



## Espacenet

**Bibliographic data: JP 8172624 (A)**

## MPEG SIGNAL DECODING AND ITS METHOD

Publication date:	1996-07-02
-------------------	------------

**Inventor(s):** WISE ADRIAN PHILIP [GB]; DEWAR KEVIN D [GB]; JONES ANTHONY MARK [GB];  
SOTHERAN MARTIN WILLIAM [GB]; SMITH COLIN [GB]; FINCH HELEN ROSEMARY [GB];  
CLAYDON ANTHONY PETER JOHN [GB]; PATTERSON DONALD WILLIAM W [GB];  
BARNES MARK [GB]; KULIGOWSKI ANDREW PETER [GB]; ROBBINS WILLIAM PHILIP  
[GB]; BIRCH NICHOLAS [GB] ±

**Applicant(s):** DISCOVISION ASS (US) +

Applicant(s): DISCOVISION ASS (US) +

**Classification:**

**International:**

G06F12/00; G06F12/02; G06F12/06; G06F13/00; G06F13/16;  
G06F13/28; G06F13/37; G06F3/14; G06F7/00; H03M7/30;  
H03M7/42; H04L1/08; H04N5/92; H04N7/26; H04N7/32;  
H04N7/50; H04N7/62; G06F12/04; (IPC-1:7): H04N5/92; H04N7/24;  
G06F12/02B; G06F12/06A; G06F13/16; G06F13/28; H04N7/26A4V;  
H04N7/26L; H04N7/26L2; H04N7/50; H04N7/50E2; H04N7/62

- European:

Application  
number:

JP19950224473 19950728

Priority number(s):

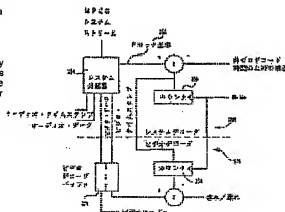
GB19940015413 19940729: GB19950011569 19950607

**Also published as:**

- EP 0695095 (A2)
- EP 0695095 (A3)
- US 5835792 (A)
- US 5829007 (A)
- US 5878273 (A)
- more

## Abstract of JP 8172624 (A)

**PROBLEM TO BE SOLVED:** To decide whether or not an image is delayed by using a 1st, counter 50 as to keep a system time in a circuit and allowing a 2nd counter to decide a display time error between the system time and the image time. A video clock reference 253 is decoded by a system separator 254, and the decoded signal is fed to a 1st counter 255 that counts a time incremented by a prescribed frequency, and also loaded to a 2nd counter 256 that counts a time incremented by a video time stamp passes through a video buffer 271, where it is delayed by a same amount as video data. They are compared next to a faster local copy of the counter or not an image is faster or slower is discriminated. The counter 255 keeps a local copy in a 1st circuit and is synchronously with the counter 258. Furthermore, the counter 258 keeps a local copy of the system time and decides a display time error between the system time by comparing the copy with the time stamp.



Last updated:  
04.04.2011 Worldwide  
Database 5.7.20: 920

特開平8-172624

(43) 公開日 平成8年(1996)7月2日

(51) Int.Cl.<sup>6</sup>H 0 4 N 7/24  
5/92

識別記号

序内整理番号

F I

技術表示箇所

H 0 4 N 7/ 13  
5/ 92Z  
H

審査請求 未請求 請求項の数29 書面 外国語出願 (全596頁)

(21) 出願番号 特願平7-224473

(22) 出願日 平成7年(1995)7月28日

(31) 優先権主張番号 9 4 1 5 4 1 3 . 5

(32) 優先日 1994年7月29日

(33) 優先権主張国 イギリス (G B)

(31) 優先権主張番号 9 5 1 1 5 6 9 . 7

(32) 優先日 1995年6月7日

(33) 優先権主張国 イギリス (G B)

(71) 出願人 591226829

ディスコビジョン アソシエイツ  
アメリカ合衆国, カリフォルニア州  
92714, アーバイン, スウィート 200, メ  
イン・ストリート 2355(72) 発明者 エイドリアン ビー. ワイズ  
イギリス国 ブリストル ビーエス16 1  
エヌエフリーチエイ ウェストボーンコ  
テージズ 10(72) 発明者 ケビン ディ. デュワー  
イギリス国 ブリストル ビーエス7 8  
エイチエイチ パークレイアベニュー 16

(74) 代理人 弁理士 伊藤 嘉昭

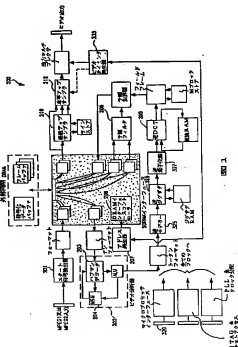
最終頁に続く

(54) 【発明の名称】 MPEG信号復号方法及び装置

## (57) 【要約】

【課題】 バイブライン処理マシンとして配置された二線式インターフェースにより相互接続された複数のステージを用いたMPEGビデオ拡張方法及び装置を提供する。

【解決手段】 制御トークン及びデータトークンがトークン形式の制御及びデータの両方を担持する単一の二線式インターフェースに渡される。トークンデコード回路がステージの内の或るものに配置され、トークンの内の或るものをそのステージに関係する制御トークンとして認識し、認識されない制御トークンをバイブラインに沿って通過させる。再整理処理回路が選択されたステージ内に配置され、認識された制御トークンにตอบสนองし、識別されたデータトークンを扱うようにそのステージを再構成する。メモリアドレス指定、共通処理ブロックを用いたデータ変換、時間同期、非同期バッファリング、ビデオ情報の記憶、並列ハフマンデコーダ、等を含むシステムを実動化するために種々の独特が支援サブ・システムの回路及び処理技術が開示されている。



## 【特許請求の範囲】

【請求項 1】 時間を同期させるための装置であって、  
提示時間を決定するタイムスタンプと、

第 1 の回路内のシステム時間を初期化するためのクロック基準と、

第 1 の回路内のシステム時間を維持するための前記クロック基準と通信する第 1 の時間カウンタと、

前記クロック基準により初期化される第 2 の回路内の前記第 1 の時間カウンタと同期した第 2 の時間カウンタであり、前記システム時間の局所コピーを維持し、前記タイムスタンプを前記第 2 の時間カウンタに比較することによって前記システム時間の局所コピーと前記システム時間との間の提示時間エラーを決定するための前記第 2 のカウンタとから成ることを特徴とする前記時間同期装置。

【請求項 2】 システムデコーダ及びビデオデコーダを同期させるための装置であって、  
システムデコーダと、

表示時間を決定するタイムスタンプと、

前記システムデコーダ内のシステム時間を初期化するためのクロック基準と、

前記システムデコーダ内のシステム時間を維持するための前記クロック基準と通信する第 1 の時間カウンタと、

前記第 1 の時間カウンタに同期した前記ビデオデコーダ内の前記クロック基準により初期化される第 2 の時間カウンタであり、システム時間の局所コピーを維持し、タイムスタンプを前記第 2 の時間カウンタに比較することによってシステム時間の前記局所コピー及び前記システム時間の間の表示タイミングエラーを決定するための第 2 の時間カウンタとから成ることを特徴とする前記同期装置。

【請求項 3】 第 1 の回路及び第 2 の回路を同期させるための装置であって、  
第 1 の回路内のシステム時間を初期化するためのクロック基準であって、システム時間を維持するために前記第 1 の回路は該クロック基準と通信する時間カウンタを有している前記クロック基準と、  
基本ストリーム時間を提供するための前記第 1 の回路内の第 1 の基本ストリーム時間カウンタとから成り、  
前記第 1 の回路はタイムスタンプを受け取るように成されており、また前記第 1 の回路は基本ストリーム時間を前記タイムスタンプに加算しシステム時間を減算することにより同期時間を生成するように成されており、  
前記第 2 の回路は前記第 1 の回路から同期時間を受取るように成されており、前記基本ストリーム時間の局所コピーを提供し、同期時間を前記基本ストリーム時間の局所コピーに比較することにより前記システム時間及び前記タイムスタンプの間のタイミングエラーを決定するための前記第 1 の基本ストリーム時間カウンタに同期した第 2 の基本ストリーム時間カウンタを有しており、

これによってタイミング・エラーを決定するために前記クロック基準信号が前記第 2 の回路に直接的に渡される必要を無くしたことを特徴とする前記同期装置。

【請求項 4】 第 1 の回路及び第 2 の回路を同期させるための装置であって、

第 1 の回路内のシステム時間を初期化するためのクロック基準と、

前記第 1 の回路はシステム時間を維持するために前記クロック基準と通信する時間カウンタを有しており、

ビデオ復号時間を提供する第 1 のビデオ時間カウンタとから成り、

前記第 1 の回路はビデオタイムスタンプを受け取り、ビデオ復号時間をビデオタイムスタンプに加算しシステム時間を減算することにより同期時間を生成するように成されており、

前記第 2 の回路は前記第 1 の回路からの同期時間を受取るように成されており、ビデオ復号時間の局所コピーを設け、前記同期時間を前記ビデオ復号時間の局所コピーに比較することにより前記システム時間及び前記ビデオタイムスタンプの間のタイミングエラーを決定するための前記第 1 のビデオ時間カウンタに同期した第 2 のビデオ時間カウンタを有しており、

これにより、タイミング・エラーを決定するために前記クロック基準信号が前記第 2 の回路に直接的に渡される必要を無くしたことを特徴とする前記同期装置。

【請求項 5】 タイミング情報を提供するのための方法であって、  
パケットヘッダー内に担持されるタイムスタンプであり、データのバケット内の最初の画面を示す前記タイムスタンプを有するビデオデータストリームを設け、  
前記パケットヘッダーから取り出されてレジスタに配置された有効タイムスタンプ情報を示すフラグを有するレジスタを設け、  
前記ビデオデータストリームから前記タイムスタンプを除去して前記レジスタに配置し、  
画面スタートに遭遇し、有効なタイムスタンプ情報が前記レジスタに含まれるか否かをフラグの状態をチェックすることにより判別するために前記レジスタの状態をその後に調べることを特徴とする前記タイミング情報検出方法。

【請求項 6】 ビデオを復号処理するための方法であって、

しきい値に対する表示時間エラーを決定し、  
その後の処理のためにビデオデータをトークンに構文解析し、  
タイムスタンプトークンが示されたか否かを判別し、  
タイムスタンプトークンをビデオ時間に比較し、  
比較値を生成してタイミングエラーの指示を決定し、  
タイミングエラーが指示された時にはしきい値と比較した時に比較値が許容可能なパラメータ内にあるか否かを

判別し、比較値が許容可能なパラメータ外にあるときにこれを表示する各ステップから成ることを特徴とする前記復号処理方法。

【請求項 7】 システムデコーダ及びビデオデコーダを使用する装置であって、MPEGシステムストリームを受け入れるように成され、ストリームからビデオデータ及びビデオタイムスタンプを分離するシステムデコーダを有し、前記システムデコーダはシステム時間を表す第 1 の時間カウンタを有しており、前記ビデオデータ及び前記ビデオタイムスタンプを受け入れるビデオデコーダを有し、前記ビデオシステムは前記第 1 の時間カウンタと同期した第 2 の時間カウンタを有し、前記ビデオデコーダは前記ビデオデータを実質的に一定の速度で受け入れ前記ビデオデータを可変の速度で出力し、ビデオタイムスタンプを渡すためのビデオデコーダバッファを有することを特徴とする前記装置。

【請求項 8】 第 1 の回路及び第 2 の回路の間のタイミングエラーを決定するための方法であって、第 1 の回路にシステム時間 (SY)、タイムスタンプ (TS) 及び基本ストリーム時間 (ET) を提供し、基本ストリーム時間 (ET)、タイムスタンプ (TS) 及びシステム時間 (SY) を使用し、式  $X = ET + TSY$  に従って同期時間 (X) を得て、同期時間 (X) を第 2 の回路に提供し、同期した基本ストリーム時間 (ET2) を生成し、同期時間 (X) を用いて式  $ET2 - X$  に従ってタイミングエラーを得る各ステップから成り、これによりシステム時間を第 2 の回路にわたすことなしに第 1 の回路が第 2 の回路に同期可能であることを特徴とする前記タイミングエラー決定方法。

【請求項 9】 第 1 の回路及び第 2 の回路の間のタイミングエラーを決定する方法であって、第 1 の回路にタイムスタンプ (TS) 及び初期時間 (IT) を提供し、タイムスタンプ (TS) 及び初期時間 (IT) を使用して、式  $X = TS - 1$  に従って同期時間 (X) を求め、同期時間 (X) を第 2 の回路に提供し、同期した基本ストリーム時間 (ET) を生成し、同期時間 (X) を使用し、式  $ET - X$  に従ってタイミングエラーを生成し、これにより第 1 の回路は時間を第 2 の回路にわたすこと無しに第 2 の回路と時間同期することが可能であることを特徴とする前記決定方法。

【請求項 10】 第 1 の回路及び第 2 の回路の間のタイミングエラーを検出するための方法であって、第 1 の回路にシステム時間 (SY)、ビデオタイムスタンプ (VTS) 及びビデオ復号時間 (VT) を提供し、

ビデオタイムスタンプ (VTS) 及びシステム時間 (SY) を使用して式  $X = VT + VTS - SY$  に従ってビデオ復号時間 (VT) を得て、同期時間 (X) を第 2 の回路に提供し、

第 1 の回路におけるビデオ復号時間 (VT) に同期した第 2 の回路におけるビデオ復号時間 (VT2) を生成し、同期時間 (X) を使用して式  $VT2 - X$  に従ってタイミングエラーを得る各ステップから成り、これによって、システム時間を第 2 の回路に渡すこと無しに第 1 の回路が第 2 の回路に時間同期可能にしたことを特徴とする前記検出方法。

【請求項 11】 メモリをアドレス指定する方法であって、可変幅データをアドレス指定するために用いられる所定の固定ビット数を有する固定幅の語を提供し、幅定義フィールド及びアドレスフィールドを有する固定幅の語を定義し、幅定義フィールドに終端マークとして作用する少なくとも一つのビットを設け、アドレスフィールドにデータのアドレスを画定する複数のビットを定義し、アドレスフィールド内のビットのサイズを可変幅データのサイズと逆の関係で変化する、幅定義フィールド内のビットの数を可変幅データのサイズに対して直接の関係で変化する、幅定義フィールド及びアドレスフィールドの幅を変化させる一方、可変幅データをアドレス指定するための固定幅語を維持する各ステップから成ることを特徴とする前記アドレス指定方法。

【請求項 12】 メモリをアドレス指定する方法であって、データをアドレス指定するために用いるための所定の固定数のビットを有する固定幅の語を提供し、アドレスフィールド及び置換フィールドを有する固定幅の語を定義し、データのアドレスを画定する複数のビットを有するアドレスフィールドを定義し、少なくとも一つの置換ビットを有する可変幅置換フィールドを定義し、置換フィールドはアドレスフィールド及び置換フィールドの間の終端マークとして作用する少なくとも一つのビットを有しており、置換フィールドを別個のアドレス指定ソースからの置換ビットを示すために使用し、アドレスフィールドの幅と置換フィールドの幅を逆に変化させる一方、可変幅データをアドレス指定するための固定幅の語を維持する各ステップから成ることを特徴とする前記アドレス指定方法。

【請求項 13】 メモリ内の可変幅データをアドレス指



定するための方法であって、  
所定幅であり部分語から成る語を有するメモリを設け、  
アクセスされるべき部分語を少なくとも有効ビット調整  
に回転し、

アクセスされた語が部分語として認識されるように語の  
残りの部分を拡張し、語の残りの部分を復元し、  
部分語が元の位置に復元されるまで語を回転する各ステ  
ップから成ることを特徴とする前記アドレス指定方法。

【請求項14】 セレクタと、  
ハフマンコード化データを受取るための一対の入力レジ  
スタであり、その両方が入力を前記セレクタに並列に方  
向付ける前記レジスタと、  
前記セレクタ及び他のROM表選択入力からの入力を受  
取るハフマン符号ROMであり、復号データ出力を提供  
する前記ROMとから成る並列ハフマンデコーダ。

【請求項15】 バスをRAMに接続するためのRAM  
インターフェースであって、  
バスから複数のデータ語を受取り、受け取ったデータ語  
をバッファリングする手段と、

前記バスから前記複数のデータ語に付随するアドレスを  
受取る手段と、

RAM内にバッファリングされたデータ語が書き込まれ  
る一連のアドレスであり、該一連のアドレスは受け取  
ったアドレスから導出される一連のアドレスを生成する  
ための手段と、

前記バッファリングされたデータ語を生成されたアドレ  
スにおいてRAM内に書き込むための手段とから成るこ  
とを特徴とする前記RAMインターフェース。

【請求項16】 バスをRAMに接続するためのRAM  
インターフェースであって、  
RAM内の所定のアドレスに記憶された複数のデータ語  
と、

バスから前記複数のデータ語に付随するRAMアドレ  
スを受け取る手段と、

前記RAM内の前記複数のデータ語をアドレス指定する  
ための一連のRAMアドレスであって、受け取ったアド  
レスから導出される該一連のアドレスを生成するための  
手段と、

前記RAMから読み出されるデータ語をバッファリン  
グするための手段と、

前記RAMから前記アドレス生成手段により生成された  
前記一連のRAMアドレスを使用して前記複数のデータ  
語を読み出し、前記データ語を前記バッファ手段に書き  
込む手段とから成ることを特徴とする前記RAMインタ  
ーフェース。

【請求項17】 フレームとして構成された符号化され  
たビデオデータのバッファリング制御するための方法で  
あって、

フレームの画面番号を判別し、  
前記フレームの所望の提示番号を決定し、

前記画面番号が前記所望の提示番号上または以降であ  
るときにバッファが使用可能であることをマークする各ス  
テップから成ることを特徴とする前記制御方法。

【請求項18】 データを変換するための装置であ  
って、  
第1のデータストリーム源を画定する第1のラッチ及び  
第2のデータストリーム源を画定する第2のラッチを有  
し、

前記第1及び前記第2のラッチは演算ユニットと通信  
し、  
前記演算ユニットはデータを転置装置に供給し、  
前記転置装置は前記データを転置して前記第2のラッチ  
に供給し、

前記第2のラッチはデータを吸収するように配置されて  
おり、  
前記第2のラッチ及び前記第1のラッチは前記第1及び  
第2のデータストリームを前記演算ユニットにインター  
リーブした状態で供給し、更に前記インターリーブされ  
た供給において前記第2のラッチは前記第1のラッチか  
らの供給に割り込まないことが画定されており、  
これによって前記第1及び前記第2のデータストリーム  
のために共通の演算ユニットが使用されることを特徴と  
する前記データ変換装置。

【請求項19】 共通の演算ユニットを使用してデー  
タを変換するための処理方法であって、

データを第1のラッチにロードし、所定のサイクル数に  
達したときにはデータを演算ユニットに送出し、第1の  
マーカビットを制御シフトレジスタにロードし、  
データを第2のラッチにロードし、第2のラッチはデー  
タを吸収するように成されており、  
第1の制御シフトレジスタが所定の状態に達し、第2の  
ラッチが所定の量のデータで満たされたときには第2の  
ラッチ内のデータを演算ユニットに送出し、  
第2のラッチが所定の量のデータで満たされていなければ  
第2のラッチからデータ送出せず、

第1のラッチがデータを受け取らないときには第2のラ  
ッチを回復させる各ステップから成ることを特徴とする  
前記処理方法。

【請求項20】 少なくとも2つのメモリアレイと、  
前記メモリアレイと通信し前記メモリアレイへのデー  
タ入力を制御する書き込み制御回路と、

前記メモリアレイと通信し、前記メモリアレイへのデー  
タ入力を制御する読み出し制御回路と、

前記メモリアレイと通信し、前記メモリアレイからのデ  
ータ出力を制御する制御回路とから成り、

前記書き込み制御回路及び前記読み出し制御回路は前記  
メモリアレイの同期した制御を可能にするため通信する  
ことを特徴とするスイングバッファ装置。

【請求項21】 メモリ内のセルを同期してアクセスす  
る方法であって、

少なくとも一対のセルを復号するデコーダを使用し、セルの一つを読み取り、他のセルに書き込む各ステップから成ることを特徴とする前記アクセス方法。

【請求項 2】 ビデオ情報を記憶するための方法であ

って、ビデオ情報を I フレーム、P フレーム、B1 フレーム及び B2 フレームのかたちで提供し、

I フレームを第 1 のフレーム記憶装置に記憶し、P フレームを第 2 のフレーム記憶装置に記憶し、第 1 の及び第 2 のフィールド記憶装置を有する第 3 の記憶装置を提供し、第 1 及び第 2 のフィールドはそれぞれ少なくとも 2 つのメモリ領域に分割されており、メモリ領域の選択された部分からの B1 フレームを第 1 または第 2 のフィールド記憶装置に記憶し、B2 フレームの一部分をメモリ領域のフレーム B1 が読み出された選択された部分に書き込み、

これによってビデオ情報を記憶するためにより少ないメモリを使用することが可能であることを特徴とする前記ビデオ情報記憶方法。

【請求項 3】 「無頓着な」処理のためのメモリであ

って、一組のメモリアドレスライン、反転アドレスライン及びデータラインから成り、前記アドレスライン及び反転アドレスラインはデータ語の形でアドレス指定された情報にアクセスするための復号フォーマットに従って接続されており、データライン上の「無頓着な」アドレス位置はアドレスライン及び前記反転アドレスラインとは接続されないことを特徴とする前記メモリ。

【請求項 4】 二次元画像に関連するデータ語を記憶し、読み出すダイナミック・ランダム・アクセスメモリ (DRAM) にアクセスする方法であり、DRAM は二つの別個のバンクを有し、各バンクはデータ語を読み出し書き込むためにページ・モードで動作可能であり、二次元画像はセルの二次元格子パターンとして構成されており、各セルは画素の M×N マトリクスを含み、各セルに関連する語はバンクの 1 ページまたはそれ以下を占める前記方法であって、

(a) 各セルに二つのバンクの内の特定の二つを割当て、その特定のセルに関連する全てのデータ語がその特定のバンクの特定の 1 ページから読み出され書き込まれるようにし、セルへのバンクの割当ては各セルがそれらも同一の行または同一の列にある境界セルとは異なるバンクに関係づけられるように成されるステップと、

(b) 画素のマトリクスから構成され、二次元格子パターンとは整列されていないけれども、二次元格子パターン内のセル内の画素と整列されているセルに関連するデータ語を読み取るステップから成ることを特徴とする前記アクセス方法。

【請求項 5】 二次元画像に関連するデータ語を記憶し読み出すダイナミック・ランダム・アクセスメモリ

(DRAM) にアクセスする方法であり、DRAM は二つの別個のバンクを有し、各バンクはデータ語、各セルが画素の M×N マトリクスを含むセルの二次元格子パターン、バンクの 1 ページまたはそれ以下を占める各セルに関連する語を書き込み出すためにページ・モードで動作可能である前記アクセス方法であって、

(a) 各セルに二つのバンクの内の特定の二つを割当て、その特定のセルに関連する全てのデータ語がその特定のバンクの特定の 1 ページから読み出され書き込まれるようにし、セルへのバンクの割当てはそれらもまた同一の行または同一の列にある境界セルとは異なるバンクに関係づけられるように行われるステップと、

(b) 画素の M×N マトリクスから構成され、二次元格子パターンと整列されていないけれども、二次元格子パターン内のセル内の画素と整列されているセルに関連するデータ語を読み出すステップから成ることを特徴とする前記アクセス方法。

【請求項 6】 前記 DRAM は第 1 及び第 2 のバンクを含み、非整列セルに関連するデータ語を読み出すステップ (b) は、

(c) DRAM の第 1 のバンクから、非整列セルに関連するデータ語を含む格子パターン内のセルの一つに関連するデータ語を読み出し、

(d) DRAM の第 2 のバンクから、非整列セルに関連するデータ語を含む格子パターン内の他のセルに関連するデータ語を読み出し、

(e) 非整列セルに関連するデータ語の全てが読み出されるまでステップ (c)、(d) を繰り返す各ステップを含むことを特徴とする請求項 6 に記載の方法。

【請求項 7】 RAM から RAM の所定の固定バースト長 N より小さな数 M の語にアクセスする方法であり、RAM は RAM からの読み出しおよび RAM への書き込みを選択的に可能化し不可能化する可能化ラインを含んでいる前記方法であって、RAM から読み出したまたは RAM に書き込むべき N 語を指定し、

N より小さな数 M の語が RAM から読み出されたかまたは RAM に書き込まれたことを判別し、M 語が RAM から読み出されたかまたは RAM に書き込まれたことが判別されたときに RAM を不可能化する各ステップから成ることを特徴とする前記アクセス方法。

【請求項 8】 RAM から RAM の所定の固定バースト長 N より小さな数 M の語を読み出す方法であり、RAM は RAM からの読み出しを選択的に可能化したまたは不可能化する可能化ラインを含んでいる前記読み出し方法であって、

RAM から読み出すべき N 語を指定し、N より小さな M 語が RAM から読み出されたことを判別し、

RAM から M 語が読み出されたことが判別されたときに

RAMを不可能化する各ステップから成ることを特徴とする前記読み出し方法。

【請求項29】 所定の固定バースト長Nより小さな数Mの語をRAMに書き込む方法であって、RAMはRAMへの書き込みを選択的に可能化した不可能化する可能化ラインを含む前記書き込み方法であって、

RAMに書き込むべきN語を指定し、RAMにNより小さなM語が書き込まれたことを判別し、

RAMにM語が書き込まれたことが判別されたときにRAMを不可能化する各ステップから成ることを特徴とする前記書き込み方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は複数のオーディオ及びビデオ信号を復号化する新たな改良システムに関し、特に、複数のMPEGオーディオ及びビデオ信号を復号化する新たな改良システムに関する。

【0002】

【従来の技術】本発明の直列パイプライン処理システムはユニークで特殊な対話型接続トークンをコントロールトークン及びデータトークンの形式で再構成可能なパイプラインプロセッサとして用いられる複数の適応伸張回路等に転送するために用いられる単一の2線バスを備えている。

【0003】米国特許5,111,292号は例えば、地上伝送用のHDTV（高品位テレビ）信号を符号化／復号化する装置を開示しており、その装置は伝送用の高及び低優先チャンネルの間に圧縮したビデオコードワードを分解する優先選択プロセッサを備えている。高品位ビデオ源信号に回答して圧縮回路は圧縮したビデオデータを示す階層コードワードCWと、コードワードCWによって表されるデータの種類の定める関連コードワードTとを供給する。コードワードCW及びTに回答する優先選択プロセッサは所定のデータブロックのビット数を計数し、各チャンネルに備えられるべき各ブロックのビット数を決定する。その後、プロセッサはコードワードCWを高及び低優先コードワード列に分解する。その高及び低優先コードワード列は画像再生に対して比較的高い重要性の高い及び低い圧縮したビデオデータに各々対応する。

【0004】米国特許5,122,875号はHDTV信号を符号化／復号化する装置を開示している。その装置は高品位ビデオ源信号に回答して圧縮したビデオデータを示す階層コードワードCWと、コードワードCWによって表されるデータの種類の定める関連コードワードTとを供給する圧縮回路を備えている。コードワードCW及びTに回答する優先選択回路はコードワードCWを高及び低優先コードワード列に分解する。その高及び低優先コードワード列は画像再生に対して比較的高い重

性の高い及び低い圧縮したビデオデータに各々対応する。高及び低優先コードワード列に回答して転送プロセッサは高及び低優先コードワードの高及び低優先転送ブロックを各々形成する。各転送ブロックはヘッダ、コードワードCW及びエラー検出チェックビットを含む。各転送ブロックは追加のエラーチェックデータを供給する前方エラーチェック回路に供給される。その後、高及び低優先データはモデムに供給され、各伝送用の搬送波に対して直角振幅変調が施される。

【0005】よって、この技術分野の当業者は従来システムの欠点を取り除く新たな改良したビデオ伸張システムが要求されていることを長く感じていた。本発明はこの要求を十分に満たすものである。本発明の実施を明確にする記載において次の項目は度々用いられ、次の用語説明によって定義される。

用語説明

ブロック：8行×8列のマトリックス画素、すなわち64のDCT（離散コサイン変換）係数（ソース、量子化、逆量子化）。

色度（成分）：ビットストリームの中で定まる形式で主要な色に関する2つの色差信号のいずれか1つを表わすマトリックス、ブロック又は信号画素。色差信号に用いるシンボルはC<sub>r</sub>及びC<sub>b</sub>である。

復号化表現：符号化形式で表される如きデータ成分。

復号化ビデオビットストリーム：この明細書で定められるような1以上の連続映像の復号化表現。

復号化順：映像が伝送され及び復号化される順番。この順番は表示順と同一である必要はない。

成分：映像を構成する3つのマトリックス（輝度、及び2つの色度）のいずれか1つからのマトリックス、ブロック又は信号画素。

圧縮：データ項目を表すために用いられるビット数への低減。

デコーダ：復号化処理手段の具体例。

復号化（処理）：入力復号化ビットストリームを読み取り復号化した映像又はオーディオサンプルを生成することの明細書で定めた処理手段。

表示順：復号化映像が表示される順番。これは、一般に、復号化映像がエンコーダの入力で存在した順番と同一である。

符号化（処理）：この明細書で定められたように入力映像又はオーディオサンプルのストリームを読み取り有効な復号化ビットストリームを生成することの明細書には特定されていない処理手段。

内部（イントラ）復号化：マクロブロック又は映像の復号化。その復号化はマクロブロック又は映像からだけの情報を使用する。

輝度（成分）：ビットストリームの中で定まる形式で主要な色に関し信号の単色表現を表わすマトリックス、ブロック又は信号画素。輝度に用いられるシンボルはYである。

ある。  
マクロブロック：映像の輝度成分の16×16の部分から生じる4つの8×8ブロックの輝度データと、2つ（4：2：0クロマフォーマット）、4つ（4：2：2クロマフォーマット）、又は8つ（4：4：4クロマフォーマット）の対応する8×8ブロックの色度データ。マクロブロックは画素データを述べるために使用され、この明細書のこの部分で定めたシタックス（syntax）のマクロブロックヘッダに定められた画素値及びその他のデータ成分の復号化表現のために用いられる。この分野の当業者にはその使用は文脈から明らかである。

動き補償：画素の予測能力を改良するために動きベクトルの使用。予測は予測エー信号を形成するために用いられる予測復号化された画素値を含む過去及び／又は未来の基準映像にオフセットを与えるために動きベクトルを使用する。

動きベクトル：現在の映像の座標位置から基準映像の座標にオフセットを与える動き補償のために用いられる2次元ベクトル。

非内部復号化：マクロブロック又は映像の復号化。その復号化はそのマクロブロック又は映像からだけでなく他の時間に生じるマクロブロック及び映像からの情報をも使用する。

画素；PEL：映像の画素

映像：信号源、復号又は復元した映像データ。信号源又は復元した映像は輝度信号及び2つの色度信号を表わす8ビット数の3つの長方形マトリックスからなる。プログレッシブビデオでは1つの映像は1つのフレームに等しいが、インターレースビデオでは1つの映像は1つのフレーム、すなわち前後関係のフレームの表フィールド又は裏フィールドに関連する。

予測：現在復号化されている画素値又はデータ成分の見込みを与えるために予測器の使用。

再構成可能な処理ステージ（RPS）：認識したトークンに応答して様々な動作を行なうためにそれ自身を再構成する段階。

スライス：連続したマクロブロック。

トークン：制御及び／又はデータ機能用の対話型接続メッセージパッケージの形状の一般的な結合ユニット  
スタートコード [システム及びビデオ]：単一の符号化ストリームに挿入される32ビットのコード。それらは符号化シタックスの構成のいくつかを識別することを含み様々な目的で用いられる。

可変長符号化；VLC：頻繁に起きるイベントに短いコードワードを割り当てたままに起きるイベントに長いコードワードを割り当てる符号化の可逆処理。

ビデオ列：1以上の連続映像。

【0006】

【課題を解決するための手段】本発明は様々な制御及び

DATAトークンを有する2線パイプラインシステムにおける使用に特に適用される新たな改良された方法及び装置を提供する。システムの主要な成分は、スタート符号デコーダと、ハフマンデコーダ及びマイクロプログラマブル状態マシン（MSM）を含むビデオ分析器と、逆離散コサイン変換（IDCT）と、運動したアドレス発生ユニットを有する同期DRAMコントローラと、適正予測回路と、アップサンプリング及びビデオタイミング発生を含む表示回路とを含む。

【0007】更に重要なことは、本発明の様々な実施例は、パイプライン処理機器として配置される2線インターフェースによって相互接続された複数のステージ（段）を利用するMPEGビデオ伸張方法及び装置を含むビデオ復号化システムの様々な態様において更なる改善のために長く存在した要求を満たす。制御トークン及びDATAトークンはトークン形式で制御及びデータ両方を運搬する1つの2線インターフェースを介して供給する。トークン復号回路は設けられたステージに適切な制御トークンとして所定のトークンを識別し、パイプラインに介して非認識の制御トークンを供給するために所定のステージに設けられる。再構成処理回路は選択されたステージに設けられ、識別された制御トークンに応答して識別したDATAトークンを処理するようなステージを再構成する。独特のサーポート割システム回路及び処理技術の幅広い変更は、メモリアドレス指定、共通処理ブロックを用いるデータ変換、時間同期、非同相タイミングバッファリング、ビデオ情報の記憶、並列ハフマンデコーダ等を含んでシステムを実施するために開示される。

【0008】例として、また必要に限定するためではなく、本発明は、様々な特徴の中に、表示時間を決定するタイムスタンプと、第1の回路内のシステム時間を初期化するクロック基準と、第1の回路内のシステム時間を維持するクロック基準との伝達をなす第1時間カウンタと、その第1時間カウンタと同期をとった第2の回路内のクロック基準によって初期化される第2時間カウンタとを有して時間の同期をとり、システム時間のローカルコピーを維持し、タイムスタンプと第2時間カウンタとを比較することによりシステム時間のローカルコピーとシステム時間との間の表示タイミングエラーを決定する装置を備えている。それは更に、表示時間を決定するためにタイムスタンプと、システムデコーダ内のシステム時間を初期化するクロック基準と、システムデコーダ内のシステム時間を維持するクロック基準との伝達をなす第1時間カウンタと、その第1時間カウンタと同期をとった第2の回路内のクロック基準によって初期化される第2時間カウンタとを用いてシステムデコーダ及びビデオデコーダとの同期をとり、システム時間のローカルコピーを維持し、タイムスタンプと第2時間カウンタとを比較することによりシステム時間のローカルコピーとシ

システム時間との間の表示タイミングエラーを決定する装置を備えている。

【0009】その他の実施例は、第1の回路内のシステム時間を初期化するクロック基準を用いて第1の回路及び第2の回路の同期をとる装置を備え、第1の回路はシステム時間を維持するためのクロック基準との伝達をなす時間カウンタを有し、その装置は更に、基本ストリーム時間を与えるために第1の回路内に第1基本ストリームタイムカウンタを含む。第1の回路はタイムスタンプを入力するように適合され、第1の回路は基本ストリーム時間とタイムスタンプとを加算しシステム時間を減算することにより同期時間を発生する。第2の回路は第1の回路からの同期時間を入力するように適合され、基本ストリーム時間のローカルコピーとを比較することによりシステム時間とタイムスタンプとの間のタイミングエラーを決定する第1基本ストリーム時間カウンタと同期した第2基本ストリーム時間カウンタを有する。このように、クロック基準信号はタイミングエラーを決定するために第2の回路に直接供給される必要はない。

【0010】他の実施例においては、第1の回路及び第2の回路の同期をとる装置は第1の回路内のシステム時間を初期化するクロック基準を有している。第1の回路はシステム時間を維持するクロック基準との伝達をなす時間カウンタと、ビデオ復号時間を与える第1ビデオ時間カウンタとを有している。その第1の回路はビデオタイムスタンプを入力する用に適合され、ビデオ復号時間をビデオタイムスタンプに加算しシステム時間を減算することにより同期時間を発生する。第2の回路は第1の回路からの同期時間を入力するように適合され、ビデオ復号時間のローカルコピーを与え、かつ同期時間とビデオ復号時間のローカルコピーとを比較することによりシステム時間とビデオタイムスタンプとの間のタイミングエラーを決定する第1ビデオ時間カウンタと同期した第2ビデオ時間カウンタを有する。よって、クロック基準信号はタイミングエラーを決定するために第2の回路に直接供給される必要はない。

【0011】本発明は、またパケットヘッド内にタイムスタンプを有するビデオデータストリームを備えることによりタイミング情報を与える方法を含み、そのタイムスタンプはデータのパケットの内の最初の映像に関している。次のステップにおいて、レジスタはパケットヘッドから取り出されレジスタ内に保持される有効なタイムスタンプ情報を示すために用いられるフラグを備える。次に、タイムスタンプはビデオデータストリームから取り去られてレジスタ内に保持される。次いで、その方法は映像開始となり、それに続いてフラグ状態をチェックすることによって有効タイムスタンプ情報がレジスタに含まれるか否かを判別するためレジスタの状態を試験する。有効タイムスタンプ情報がレジスタ内に含まれるな

らばタイムスタンプは映像開始に反応して発生され、そして、そのタイムスタンプはデータストリーム内に戻される。

【0012】他の実施例は上記したように、基本ストリーム時間カウンタが16ビットに制限される装置を含む。同様に、上記したように、基本ストリームデコーダ内に設けられた第2基本ストリーム時間カウンタが16ビットに制限される装置が備えられている。更に、上記したように、同期時間は基本ストリームデコーダを制御するために16ビットにされる装置が備えられている。

【0013】また、本発明はビデオを復号し、閾値に対する表示時間エラーを決定する処理を有する。それは更なる処理を行い、タイムスタンプトークンが示されているか否かを判別し、そのタイムスタンプトークンをビデオデータと比較するためにトークン内のビデオデータを分析し、タイミングエラー量を判別するために比較した値を発生する。次に、閾値に対して比較されたときに比較した値がタイミングエラーが示されたときの許容パラメータ内にあるか否かが判別され、その比較した値が許容パラメータ以外にあるときを示す。

【0014】代替実施例はシステムデコーダとビデオデコーダを用いる装置を含む。そのシステムデコーダはMPEGシステムストリーム及び多重化されたビデオデータとそのストリームからのビデオタイムスタンプとを受け入れるように適合される。そのシステムデコーダはシステム時間を示す第1時間カウンタを有する。ビデオデコーダはほぼ一定のレートでビデオデータを受け入れ、かつ可変レートでそのビデオデータを出力し、ビデオタイムスタンプを供給するビデオデコーダバッファを有する。ビデオデータから映像を復号するビデオデコーダは復号された映像のためのビデオタイムスタンプを適切な表示時間を決定するように第2時間カウンタと比較する。第1の回路にシステム時間(SY)、タイムスタンプ(TS)及び基本ストリーム時間(ET)を与えることにより第1の回路と第2の回路との間の時間エラーを決定し、基本ストリーム時間(ET)、タイムスタンプ(TS)及びシステム時間(SY)を用いることによって同期時間(X)を得て、式 $X = E + T + S - SY$ に応じて第2の回路に同期時間(X)を与えかつ同期した基本ストリームタイム(E T 2)を発生し、同期した時間を用いることによって時間エラーを得る方法が備えられ、よって、第2の回路にシステム時間を供給することなく、式 $E T 2 - X$ に応じて第1の回路は第2の回路に同期した時間に行うことができる。

【0015】第1の回路と第2の回路との間の時間エラーを決定する他の方法は次のステップを有している。すなわち、第1の回路にタイムスタンプ(TS)及び初期時間(IT)を与え、タイムスタンプ(TS)及び初期時間(IT)を用いることによって同期時間(X)を得て、式 $X = T - S - 1$ に応じて第2の回路に同期時間

(X)を与えと共に同期した基本ストリーム時間(ET)を発生し、同期した時間(X)を用いて式 $ET-X$ に応じて時間エラーを得る。このように、第1の回路は、第2の回路にシステム時間を供給することなく、第2の回路に同期した時間に行うことができる。

【0016】更に、第1の回路と第2の回路との間の時間エラーを決定する他の方法は次のステップを有している。すなわち、第1の回路にシステム時間(SY)、ビデオタイムスタンプ(VTS)及びビデオ復号時間(VT)を与え、ビデオ復号時間(VT)、ビデオタイムスタンプ(VTS)及びシステム時間(SY)を用いることによって同期時間(X)を得て、式 $X=VT+VTS-SY$ に応じて第2の回路へ同期時間(X)を与えと共に第1の回路内のビデオ復号時間(VT)に同期した第2の回路内のビデオ復号時間(VT2)を発生し、同期した時間(X)を用いて式 $VT2-X$ に応じて時間エラーを得る。よって、第1の回路は、第2の回路にシステム時間を供給することなく、第2の回路に同期した時間に行うことができる。

【0017】本発明においては、並列ハフマンデコーダブロックは、ハフマン符号化可変長符号(Huffman coded Variable Length Codes:VLCs)及び固定長符号(Fixed Length Codes:FLCs)を復号化し、分析マイクロプロセッサ状態マシン(Microprogrammable state machine:MSM)の制御の元でトークンを介して供給し、高い処理能力を維持する。

【0018】本発明の実施例においては、符号テーブル索引技術は動作要求を達成するためハフマン符号を復号化し、実際に不規則又は非標準である第2のMPEG-2変換係数テーブルを処理するために用いられる。本発明の実施は、外部コントローラの助けなく単一のサイクル内にストリームからの非常に複雑な成分を復号化することを容易にする。そのような複雑な成分の例はエスケープ符号化係数(Escape-coded coefficients)、イントラDC値(Intra-DC values)及び動きベクトルデルタ(Motion Vector delta)であり、それらの全ては結合したVLC/FLC成分としてストリーム内に与えられる。

【0019】VLCを復号化するため入力は、最上位及び最下位データを処理する2つの入力データレジスタに先ず読み込まれる。セレクトはROM入力で次のVLCの始めを割り当てるために用いられる。よって、非常に早いVLCを復号するためにセレクトは59ビット入力の最初の28ビットを出力し、それらの最初の16ビットはハフマン符号ROM302に供給される。それに続くVLCに対してセレクトはここまで符号化されたビットの合計の計数値に応じた入力を効果的にシフトす

る。その計数値は復号化されるままに、実行合計に各VLCのサイズを加算することによって保持される。様々なワード幅は、復号することができる最大符号化サイズ(28ビットのMPEG-1エスケープ符号化係数)の結果と、16ビットである最大VLCサイズ(DCT係数テーブル)とある。

【0020】「テーブル選択」入力はMPEGによって要求される様々な異なるハフマン符号テーブル間の選択のために用いられる。ROMはセレクト/シフトで制御されるアドレスを有している。ROMはVLCテーブル索引計算を行い、それに続いてデータ索引動作(Indextooperation)が復号化データを生成する。

【0021】索引計算は、与えられたデータを生成するハフマン符号を処理するために実行される「ドントケア(don't care)」マッチングを有する内容アドレス指定可能メモリ(CAM)の動作である。索引生成は(むしろアルゴリズム的に)テーブル索引(ルックアップ)方法で行われるので、標準のテーブルを処理する制限はない。

【0022】本発明において、ROMのアドレスは2つのフィールド内にある。大きいフィールドは復号化されるべきビットパターンであり、小さいフィールドは調べられるべきハフマン符号を選択する。完全なMPEG符号テーブルに加えて、ROMは所定の符号テーブル内に存在する不当なVLCパターンを識別する入力を有する。

【0023】本発明の他の実施例において、プロシージャ(処理)は、可変幅データをアドレス指定するために用いられるべき固定ビット数を有し、かつ幅設定フィールド及びアドレスフィールドを有する固定幅を有するワードを与えるために用いられる。また、データをアドレス指定するために用いられるべき固定ビット数を有し、かつ置換フィールド及びアドレスフィールドを有する固定幅ワードを有するメモリをアドレス指定するプロシージャと、状態マシン及び演算コアを含んでメモリをアドレス指定する装置とがある。

【0024】メモリをアドレス指定するプロシージャは、可変幅データをアドレス指定するために用いられるべき所定の固定ビット数を有する固定幅ワードを供給し、幅設定フィールドと、終端マーカとして作用するため少なくとも1ビットを幅設定フィールドに与えるアドレスフィールドとを備えた固定幅ワードを設定し、データのアドレスを設定する複数のビットからなるアドレスフィールドを設定し、可変幅データのサイズと反比例してアドレスフィールド内のビットサイズを変化させ、可変幅データのサイズに比例して幅設定フィールド内のビット数を変化させ、可変幅データをアドレス指定する固定幅ワードを保持する一方、幅設定フィールド及びアドレスフィールドの幅を変化させることによって特徴づけ

られる。

【0025】メモリをアドレス指定するプロシージャは、また、データのアドレスを設定する複数のビットからなるアドレスフィールドを設定し、少なくとも1つの置換ビットからなる可変幅置換フィールドを設定し、その置換フィールドはアドレスフィールドと置換フィールドとの間に終端マークとして作用するために少なくとも1ビットを有し、分離のアドレス指定源からの置換ビットを示すために置換フィールドを用い、可変幅データをアドレス指定する固定幅ワードを保持する一方、アドレスフィールドの幅と置換フィールドの幅とを逆に変化させる。

【0026】本発明においては、メモリ内の可変幅データをアドレス指定する処理は、所定の幅でかつ部分ワードからなるワードを有するメモリを備え、最下位ビット調整にアクセスさせるために部分ワードを循環させ、アクセスしたワードが部分ワードとして認識されるようにワードの残り部分を拡張し、ワードの残り部分を元に戻して部分ワードが元の位置に戻るまでワードを循環させることによって特徴づけられる。

【0027】本発明は、メモリをアドレス指定する方法及び装置を含み、ワードは可変幅データをアドレス指定するために用いられるべき固定ビット数を有し、かつ幅設定フィールド及びアドレスフィールドを有する固定幅で与えられる。更に、データをアドレス指定するために用いられるべき固定ビット数を有し、かつ置換フィールド及びアドレスフィールドを有する固定幅ワードを有するメモリをアドレス指定するプロシージャが用いられる。

【0028】本発明は、RAMからそのRAMの所定固定バースト長Nより少ない数Mのワードをアクセスする方法であって、そのRAMはRAMからの読み出し及びRAMへの書き込みを選択的に可能及び不可能にするインエプラインを含み、その方法は、RAMから読み出され又はRAMへ書き込まれるNのワードを配列し、Nより小さいMのワードがRAMから読み出され又はRAMへ書き込まれているときを判別し、MのワードがRAMから読み出され又はRAMへ書き込まれたときRAMを不動作にさせることからなる。

【0029】本発明は、2次元の映像と対応したデータワードを記憶し及び読み出すためにダイナミックランダムアクセスメモリ(DRAM)をアクセスする方法であって、そのDRAMは2つの分離したバンクを有し、各バンクはデータワードを読み出す及び書き込むページモードを動作することができ、2次元映像はセルの2次元グリッドパターンの中に構成され、各セルは画面のMXNマトリックスを含み、各セルと対応したワードはバンクの1ページ以下を占め、その方法は、(a)2のバンクのうちの特定の1つと対応した全てのデータワードがその特定のバンクの特定のページについて読み出され及

び書き込まれるように各セルに2のバンクのうちの特定の1つを割り当て、同一の行又は同一の列のいずれかにある隣り合うセルよりも異なるバンクと各セルが対応するようにセルへのバンクの割り当てがされてお

(b) 画面のマトリックスからなるセルと対応し、2次元グリッドパターンと位置合わせせず、2次元グリッドパターン内のセルの画面と位置合わせされたデータワードを読み出し、(c) 2次元グリッドパターン内のセルが位置合わせされていないセルと対応したデータワードを含むことを識別し、(d) 位置合わせされていないセルと対応したデータワードを含むように識別されたグリッドパターンの中のセルの一方と対応するデータワードをDRAMの第1のバンクから読み出し、(e) 位置合わせされていないセルと対応したデータワードを含むように識別されたグリッドパターンの中のセルの他方と対応するデータワードをDRAMの第2のバンクから読み出し、(f) 位置合わせされていないセルと対応したデータワード全てが読み出されるまで(e)及び(f)を繰り返すことである。

【0030】本発明は、RAMへのバスを接続するRAMインターフェースを提供し、分離アドレス発生器はアドレスを発生し、RAMインターフェースはRAMをアドレス指定するために必要である。そのアドレス発生器は2線インターフェースを介してRAMインターフェースと伝送転送する。本発明はフレーム又はフィールドとして構成された符号化ビデオデータのバッファリングを制御する方法を含む。この方法は各入力符号化フレームの映像番号を決定し、常に期待の表現番号を決定し、その映像番号がその表現番号以降のとき準備としていずれかバッファにマークをつけることを含む。

【0031】よって、デオデータを復号化するシステムの設計、開発及び利用と関係したることについて、本発明の様々な特徴によって達成されるように拡張した実施のための必要を長く認める。本発明のその他の目的及び利点は図面と共に必要とされる次の詳細な説明から明らかになる。

#### 【0032】

【実施例】以下の「本発明の詳細な説明」は次の章を含んでいる。

1) メモリアドレス指定についての本発明の詳細な説明  
固定幅ワード内の可変長フィールド  
アドレス指定を換えるための可変長フィールドを備えた固定幅ワードの使用

アドレス置換  
固定幅ワードを備えた可変幅データのアドレス指定  
マイクロコード化可能な状態マシン構造  
演算コア

2) 共通処理ブロックを用いたデータ変換の発明の詳細な説明

発明の理論背景

- 3) 時間同期についての発明の詳細な説明  
 4) 非同期のスイングバッファリングについての発明の詳細な説明  
 5) ビデオ情報の記憶についての発明の詳細な説明  
 6) 並列ハフマンデコードについての発明の詳細な説明

ハフマン符号 ROM  
 処理能力の最大化  
 FLC 及びトークン  
 実施

- 7) 更に詳細な説明  
 発明の詳細な説明

本発明の最も一般的な特徴の図示の実施例を、特に図面の図 1 を参照して説明するが、本発明の好ましい実施例 200 からデータの流は示されている。本発明の実施例は、様々な制御及び D/A トークンを有する 2 線パイプラインシステムを使用している。そのシステムの主要な要素はスタートコード検出器 201 と、ハフマンデコード 203 及びマイクロプログラムブル状態マシン (MSM) 204 を組み込んだビデオ分析器 202 と、逆離散コサイン変換 (IDCT) 205 と、関連アドレス発生ユニット 207 を備えた同期 DRAM コントローラ 206 と、適切な予測回路 208 と、アップサンプリング 210、211 及びビデオタイミング生成回路 208 を含む表示回路 209 である。

【0033】本願は Discovision Associates によって 1994 年 3 月 24 日に「Video Decompression」という名称の英国特許出願 9405914、4 号において開示された技術内容に類似しており、その英国特許出願は本願において参照によって特に組み入れられる。上記のことに応じ、本発明の特定の態様、特徴及びサブシステムの領域は以下により詳細に言及される。図面では同様の参照符号は様々な図面中の同様の又は対応する部分を示す。

【0034】メモリアドレス指定についての発明の詳細な説明

本発明によるメモリをアドレス指定する方法及び装置をここに述べる。特に、本発明は固定幅ワードを備えた可変幅ビットフィールドを拡張するために提供する。更に、本発明は固定幅ワードを備えた可変幅データをアド \*40

\* レス指定する方法を提供する。様々な実施例では可変幅ビットフィールドはワードに置き換えられるべきビットを特定するため、又は固定幅ワードを備えた可変幅データをアドレス指定することにおいてワードの不使用部分を特定するために用いられる。更に、本発明のシステムは演算コアを有するマイクロコード化可能な状態マシンを含む。

【0035】マイクロコード化可能な状態マシンは多様な及び/又は複雑な計算の必要があるという設計問題を解決するために用いられる。そのような設計例は、アドレス発生、ストリーム分解及び符号化、フィルタタップ係数計算を含んでいる。この点については、アドレス指定は (1) ワードの可変幅部分をアクセスするために可変長アドレス、(2) アドレス代用という 2 つの異なる特徴を対処しなければならない。本発明においては、64 × 32 ビット構成を有する RAM は 64 × 32 ビット、128 × 16 ビット、256 × 8 ビット、512 × 4 ビット、1024 × 2 ビット又は 2048 × 1 ビット形式を有する部分ワードにアドレス指定され得る。

【0036】固定幅ワード内の可変長フィールド

多くのアプリケーションでは、置換、可変幅データアドレス指定、又はワードの他の部分の圧縮等の動作のために (フィールドとして知られるべき) ワードの可変部分を定めることは有用である。ワードの可変部分を定める従来の方法はワード内のフィールドの幅を特定する情報のワードを備えることである。本発明ではワード内の情報を符号化する方法を述べる。本発明の方法は、ワードの全体の定義でビットを節約すること、符号化したワードの復号化を簡単にすること及び何が符号化されているかを更に直感的に与えることの利点を有している。更に、可変幅フィールドがそのワード内で調整される最上位又は最下位ビットを含むならばこの符号化方法を適用することができる。

【0037】よって、表 1 は 8 ビットのワード内に定められ最小位ビット調整される可変幅フィールド (「F」で示す) の 2 つの例を示している。「W」はワードの他のポテンシャルフィールドを記している。

【0038】

【表 1】

ビット番号 (16 進)	7	6	5	4	3	2	1	0
固定ワード	W	W	W	F	F	F	F	F
	W	W	W	W	W	W	F	F

表 2 は 2 進数でフィールドの最大幅を特定するように十分な追加のビットを用いて表 1 に示されたフィールドを符号化する従来の方法を示している。「X」を記したビットは特定されない。すなわち、それらの値は重要では

ない。この方法はビットの使用において明らかに非効率であり、更に、本発明より直感的な形ではない。

【0039】

【表 2】



21

2

ビット番号 (16進)	7	6	5	4	3	2	1	0	フィールド定義		
固定ワード	W	W	W	X	X	X	X	X	1	0	1
	W	W	W	W	W	W	X	X	0	1	0

本発明による新たな方法は、ワード内のフィールドを定める。この方法は継続マーカ及び終端マーカを用いることによりフィールドを定める。そのフィールドはそのフィールドの一端から終端マーカまでに続く継続マーカ列として特定される。しかしながら、ゼロ長フィールドの場合には終端マーカはワード端で与えられる。継続マーカ及び終端マーカの両方とも単一ビットであり、コンプリメンタリ(相補の関係)でなければならない。加えて、フィールドはワードのいずれか端において調整され\*

\* なければならない。よって、フィールドを符号化する本発明の方法は元のワード幅より1ビットだけ余分の幅を必要とする。

【0040】その新たな方法による表1に示したフィールドの符号化は表3に示したようになる。この例において、継続マーカは「1」であり、終端マーカは「0」である。その例のフィールドは最小位ビット調整される。

【0041】

【表3】

ビット番号 (16進)	7	6	5	4	3	2	1	0
固定ワード 継続マーカ=1; 終端マーカ=0.	W	W	W	0	1	1	1	1
	W	W	W	W	W	0	1	1

よって、本発明の符号化方法の利点は次の如きである。

1. 符号化の必要なビット数が減少する。
2. 通常要求される表1, 2に示した「フィールド設定」の「Xを1」とする復号のための要求が2<sup>2</sup>のうちの1の形に既にある符号化では本来的であるので復号化処理における簡略化が要求される。
3. 定められたフィールドを容易に識別できるように符号化が直感的である。

※

【0042】更に、本発明による符号化方法の使用では、表3の符号化と同様の表4の符号化のように、終端マーカと継続マーカとを逆に用いることができる。「1」又は「0」の使用は本願において交換して用いることができる。

【0043】

【表4】

ビット番号 (16進)	7	6	5	4	3	2	1	0
固定ワード 継続マーカ=1; 終端マーカ=0.	W	W	W	1	0	0	0	0
	W	W	W	W	W	1	0	0

上記したように、符号化されたフィールドはワードのいずれかの端に調整されなければならない。表5は最上位で調整されたフィールドを示している。すなわち、それは最下位ビットに位置調整されたフィールドと同様に符号化され、符号化は最上位ビット (以下、MSBと称す) から最下位ビット (以下、LSBと称す) に向かう\*

★ 最初の終端マーカまでのフィールドにおいて行なわれる。表5に示したフィールドの符号化は表6に示されている。

【0044】

【表5】

ビット番号 (16進)	7	6	5	4	3	2	1	0
固定ワード	F	F	F	F	F	W	W	W
	F	F	W	W	W	W	W	W

【0045】

☆ ☆ 【表6】

ビット番号 (16進)	7	6	5	4	3	2	1	0	
固定ワード 継続マーカー1; 終端マーカー0.	1	1	1	1	1	0	W	W	W
	1	1	0	W	W	W	W	W	W

更に、ワードの最下位端及び最上位端から同意にフィールドを符号化しても良い。例えば、表7に示した2つのフィールドは上記したように各フィールド毎に1ビット

の追加で表8に示したように符号化することができる。

【0046】

【表7】

ビット番号 (16進)	7	6	5	4	3	2	1	0
固定ワード	F	F	F	F	W	W	F	F
	W	W	W	W	F	F	F	F

【0047】

\* \* 【表8】

ビット番号 (16進)	7	6	5	4	3	2	1	0
固定ワード	1	1	1	1	0	W	W	0
総括 マーカー=1; 終端 マーカー=0.	0	W	W	W	0	1	1	1

# アドレス置換のための可変長フィールドを備えた固定ワードの使用

他の値によってメモリアドレスの部分を置換することが有益な場合がある。この場合、データに従ったアドレスを構成することができる。アドレスの部分が何と置換されるのか特定するためにメモリのアドレスに本発明の符号化方法を適用することができる。最下位ビット調整の可変長フィールドがアドレスに用いられるならば、置換フィールドを定めることができる。例えば、12ビットのアドレス「0baaaaaaaaaaaaaa」が12ビット20

※ットの値「0bcccccccccccc」によって5つが最小位ビット側で置換されるように符号化されるならば、その12ビットのアドレス「0baaaaaaaaaaaaaa」は「0baaaaaaaaa01111」となり、アドレス「0baaaaaaaaaaccccc」が発生する。表9は12ビットアドレスへの置換のための符号化を示している。

【0048】

【表9】

表9 アドレス置換

置換ビット番号	B	A	9	8	7	6	5	4	3	2	1	0
0	a	a	a	a	a	a	a	a	a	a	a	1
1	a	a	a	a	a	a	a	a	a	a	0	1
2	a	a	a	a	a	a	a	a	a	0	1	1
3	a	a	a	a	a	a	a	a	a	0	1	1
4	a	a	a	a	a	a	a	a	0	1	1	1
5	a	a	a	a	a	a	a	0	1	1	1	1
6	a	a	a	a	a	a	0	1	1	1	1	1
7	a	a	a	a	a	0	1	1	1	1	1	1
8	a	a	a	a	0	1	1	1	1	1	1	1
9	a	a	a	0	1	1	1	1	1	1	1	1
10	a	a	0	1	1	1	1	1	1	1	1	1
11	a	0	1	1	1	1	1	1	1	1	1	1
12	0	1	1	1	1	1	1	1	1	1	1	1

# 固定ワードを備えた可変長データのアドレス指定

本発明の一実施例はその全幅で又は全幅までの2°の幅にアクセスすることができるメモリのアドレス指定用である（それらの小さいワードは部分ワードと呼ばれる）。よって、本発明の可変フィールド符号化がこのメモリに対しアドレス指定し、メモリにそれらアドレスの索引を付けるためにどのように用いられるのかを示す。

【0049】64×32ビットにアクセスするために、32、16、8、4、2及び1ビットの幅において登録ファイルは異なるアドレス長を必要とする。すなわち、この実施例のものは64×32ビット、128×16ビット、256×8ビット、512×4ビット、1024×2ビット又は2048×1ビットとしてアクセスする

ことができる64×32ビットメモリである。64×32ビットの位置のうちの1つをアドレス指定するために5ビットが必要であり、2048×1ビットの位置のうちの1つをアドレス指定するために12ビットが必要である。よって、アドレスは可変長にすることができ、実際には、アドレスの幅はメモリのアドレスフォーマットを特定する。アドレスを圧縮しその幅を定める最上位調整の可変フィールドを用いることにより固定ワード幅内でアドレスを定めることができる。これを表10に示す。

【0050】

【表10】

表 10 可変幅アドレス指定

データ幅	A	9	8	7	6	5	4	3	2	1	0
1	1	a	a	a	a	a	a	a	a	a	a
2	0	1	a	a	a	a	a	a	a	a	a
4	0	0	1	a	a	a	a	a	a	a	a
8	0	0	0	1	a	a	a	a	a	a	a
16	0	0	0	0	1	a	a	a	a	a	a
32	0	0	0	0	0	1	a	a	a	a	a

アドレスの索引を付けるためには、アドレス置換のために上記した同じ方法を用いてその部分を置換することができる。アドレスの置換部分（フィールド）については表 10 に示したそれらの最初に挿入される最小ビット調整の可変長フィールド（継続マーク「1」、終端マーク「0」）によって定めることができる。表 11 は例として 8 ビットのワードのアドレスを用いて、置換される\*

表 11 アドレス置換

置換ビット番号	A	9	8	7	6	5	4	3	2	1	0	W
0	0	0	0	1	a	a	a	a	a	a	a	0
1	0	0	0	1	a	a	a	a	a	a	a	0
2	0	0	0	1	a	a	a	a	a	a	0	1
3	0	0	0	1	a	a	a	a	0	1	1	1
4	0	0	0	1	a	a	a	a	0	1	1	1
5	0	0	0	1	a	a	a	0	1	1	1	1
6	0	0	0	1	a	a	0	1	1	1	1	1
7	0	0	0	1	a	0	1	1	1	1	1	1
8	0	0	0	1	0	1	1	1	1	1	1	1

実際には、置換コードは既に符号化されたアドレスの最初に挿入される。この符号化からはイリガルアドレス（認められないアドレス）、ほとんど 0 x 0 0 0 0 及び 0 x 3 f f f であることがわかる。この場合、「0」は 8 ビットより大きな置換を阻止するため下位 9 ビットにあり、上位 6 ビットの「1」は許容アクセス幅を特定する。それらのエラーの 1 つが検出されたならば、アクセスは設定されないが、登録ファイルの内容は影響されない。

【0052】本発明による、アドレス指定登録ファイル内の部分ワードをアクセスするシステムについて以下に示す。従来のメモリ回路は、そのメモリが全幅で常にアクセスされる必要があること、符号拡張を含んでよい。拡張は演算モードに従っている。部分ワードがメモリに入力され書き戻されたとき、それは循環した完全なワード（逆循環されアレイに書き込まれた）に重畳される。図 3 は 32 ビットワードの第 4 の 4 ビットワードに 4 ビットの部分ワードをアクセスするステップを示している。

\* べき最小ビットの数をどのように定めるのか示している。付加される最小ビットは置換の印（「W」で記した）である。置換のため固定幅ワードの通常の場合を図 2 に示す。

【0051】

【表 11】

【0053】図 3 の 1 列の 2 1 3 にハッチングで示した 4 ビットワード等の部分ワードをアクセス又は読み取るために、全幅のワードは 2 列の 2 1 4 のように L S B に部分ワードを位置させるために循環させる必要がある。3 列の 2 1 5 に示したように、4 ビットワードは完全な 32 ビットワードを作り出すために拡張される。このワードは直ちにアクセスすることができる。

【0054】図 3 に示したように、書き戻されるために、選択された全幅のワードは、2 列の 2 1 4 に示したワードに重畳されている元の部分ワードの幅に切り詰められる。L S B の位置でこれは 4 列の 2 1 6 に示されている。その結果としてのワードは読み取ったワードの中の元の位に戻さ、これは 5 列の 2 1 7 に示されている。この完全なワードを登録ファイルに書き戻すことができる。

【0055】よって、次のリストは図 3 のステップを簡単に説明している。

1. 完全なワードをメモリから読む。
2. 右へ 12 ビットの循環により L S B に部分ワードを置く。
3. 完全なワードに拡張し、そして出力へ供給する。
4. 入力した部分ワードを（2）からの循環した完全なワードに重畳する。
5. 左へ 12 ビットの循環により書き込まれるべき元の

状態に完全なワードを置く。

【0056】上記のアクセスは図4に示したメモリのデータ流れ構造を提案する。構造における番号は上記のリストの番号及び図3の列番号と対応する。メモリアドレスは上記の構造を制御するために復号される必要がある。アドレスの幅におけるMSBがメモリについて同一の位置にあることが認識されるべきである。復号したアドレスの始めの6ビットは32ビットワードのアドレスであり、残りはビットアドレスである。よって、(置換と並行した)復号段は最上位の終端マーカの位置を検出することによってアドレス幅設定可変フィールドを復号する。これはアドレスのMSBの位置が調整されることである(LSBでゼロにシフトする)。始めの6ビットをメモリの32ビットワードの列アドレスとして直接利用することができる。下位の5ビットについては(図4から分かるように、)両方のシフトを制御するために用いることができる。なぜなら、例えば、元の32ビットアドレスは「0b000000」(それらはアドレスがMSB調整されたときシフトされる)のシフトを常に有しているからである。同様に、16ビットアドレスは「0b x00000」のシフト、すなわち、0又は16ビットシフトを有することができ、1ビットアドレスは「0b x x x x」のシフト、すなわち、0から31ビットシフトを有することができる。拡張器及び入力マルチプレクサは各々出力ワードをマスキアウトし、適当な位置に入力ワードを重畳するようにアクセス幅復号によって制御される。図5に復号のブロック図を示している。幅及び置換のための2つの可変幅フィールドの復号を並行にかつ独立して行うことができることが分かる。

【0057】図2は下位の2列に示されたように可変幅データのアドレス指定及び置換のために固定幅ワード13ビットの長さの例を示している。その例では、8ビットワードは位置「0b1101ssss」にアドレス指定され、ここで、「ssss」は他のアドレス源から置換される。

#### マイクロコード化可能な状態マシン構造

本発明によれば、メモリアドレスへの置換及びメモリの可変幅アクセスは、図6に示された構成のマイクロコード化可能な状態マシンの動作において共に行なわれる。その構成は、マイクロコード指令と呼ばれる制御信号のワードワードを介して演算コア219によって制御される状態マシン218の1つである。演算コア219は状態フラグ及びいくつかのデータを順に状態マシン218へ供給する。

【0058】本発明によれば、状態マシン218はマイクロコード指令のリストを備えたメモリを有している。従来のマイクロコード化可能な状態マシンでは、マイクロコード指令のリストを介して進行する1つの指令から他の指令にジャンプすることが起きる。ジャンプアドレスは図7に示した形式である。置換された値は図6及

び図8に示したように演算コア219から供給される。これはマイクロコードプログラム内に「ジャンプテーブル」を形成させる。よって、ジャンプが例えば、3ビットの置換でされるならば、演算コアからの値によるジャンプできる8つの連続的な位置がある。すなわち、それはプログラマブルジャンプとなる。

#### 【0059】演算コア

演算コア219は、図8に示したように、登録ファイル(Register file)221と呼ばれるメモリ、演算及び論理ユニット(ALU)222、入力ポート223及び出力ポート224を備えている。それらの素子は複数のバス及び複数のマルチプレクサを介して接続されている。上記したように、それらの素子及びそれらの接続をなすマルチプレクサは、状態マシン218から発せられるマイクロコード指令によって制御される。ALU222及びポート223、224は従来と同様であるが、登録ファイル221は、可変幅索引付きのアクセスを可能にするメモリである。登録ファイル221へのアドレスはマイクロコード指令の中に直接符号化される。

【0060】登録ファイルへのアドレス指定の方法を用いることの利点は沢山ある。第一に、アプリケーション内の位置はメモリの全幅(この場合32ビット)である必要はない。全幅の位置を用いるために装置の動作に影響がないが、メモリの位置を非常に無駄に使う。メモリの位置数を最小にすることはメモリーで使用される領域を最小にし、よって、登録ファイル内の収納容量を最小にする。第二に、メモリアccessの可変幅と組み合わせる索引を付けることは可変幅の位置の全てのステップを可能にする。1ビットの場合にこれは長い除算及び乗算の的確な実行を可能にする。

【0061】よって、まとめとして次のステップを有するメモリアドレス指定処理が述べられている。

(1) 可変幅データをアドレス指定するために用いられるべき所定の固定ビット数を有する固定幅ワードを供給し、(2) 幅設定フィールドと、終端マーカとして作用するため少なくとも1ビットを幅設定フィールドに与えるアドレスフィールドとを備えた固定幅ワードを設定し、(3) データのアドレスを設定する複数のビットからなるアドレスフィールドを設定し、(4) 可変幅データのサイズと反比例してアドレスフィールド内のビットサイズを変化させ、可変幅データのサイズに比例して幅設定フィールド内のビット数を変化させ、可変幅データをアドレス指定する固定幅ワードを保持する一方、幅設定フィールド及びアドレスフィールドの幅を変化させる。加えて、次のステップを有するメモリアドレス指定処理が述べられている。

【0062】(1) データをアドレス指定するために用いられるべき所定の固定ビット数に有する固定幅ワードを供給し、(2) アドレスフィールドと置換フィールド

とを備えた固定幅ワードを設定し、(3) データのアドレスを設定する複数のビットからなるアドレスフィールドを設定し、(4) 少なくとも1つの置換ビットからなる可変幅置換フィールドを設定し、(5) その置換フィールドはアドレスフィールドと置換フィールドとの間に終端マークとして作用するために少なくとも1ビットを有し、(6) 分離のアドレス指定源からの置換ビットを示すために置換フィールドを用い、可変幅データをアドレス指定する固定幅ワードを保持する一方、アドレスフィールドの幅と置換フィールドの幅とを逆に変化させる。更に、メモリ内の可変幅データをアドレス指定する処理は次のステップを有するように述べられている。

【0063】(1) 所定の幅でかつ部分ワードからなるワードを有するメモリを備え、(2) 最下位ビット調整にアクセスさせるために部分ワードを循環させ、(3) アクセスしたワードが部分ワードとして認識されるようにワードの残り部分を拡張し、(4) ワードの残り部分を元に戻して部分ワードが元の位置に戻るまでワードを循環させる。

【0064】共通処理ブロックを用いたデータ変換の発明の詳細な説明

本発明の実施例は周波数から時間表示に信号を変換する方法及びその変換を実施するデジタル回路構成に関する。情報内容及び変換速度の両方を増加することは電気通信の分野における共通の目的である。しかしながら、送信した信号を処理しなければならぬ送受信端末でハードウェアのように、各通信媒体は送信速度の限界を有している。例えば、電信キーをたたくことより郵便文書をタイプし読み取ることの方が速いけれども電信線は一般に郵便より情報送信では大変に速い媒体である。

【0065】送信した情報を符号化する方法は、情報を運ぶことができる速度を制限する。例えば、長い電文メッセージは同一の情報内容の簡潔なメッセージより転送するためには長くなる。よって、大きな送信及び受信速度では、できるだけ沢山送信されるようにデータを圧縮することにより得て、そして高速送信媒体を用いてできるだけ速く両端末でデータを処理することができる。それはシステムにおける障害の減少又は除去を意味する。

【0066】大きなデータ量の高速送信を主に行なう1つの応用はデジタルテレビジョンの分野にある。従来のテレビジョンシステムはテレビジョン画面に表示される各ライン内の画素(ピクセル)の輝度及び色を制御するためアナログの無線信号及び電気信号を使用するが、デジタルテレビジョン送信システムは各画素毎の輝度値及び色値に対応する2進数にアナログ信号を変換することにより画像のデジタル的な表現を生成する。最近のデジタル符号化方式及びハードウェア構成は一般に、従来のアナログ送信システムより更に高い情報送信レート可能にする。デジタルテレビジョンでは、従来

のアナログのものよりもより高解像度でより生に近い画像を達成することができる。いわゆる高品位テレビジョン(HDTV)システムを含むデジタルテレビジョンシステムは、工業化された世界の大部分では次の10年間は従来のアナログテレビジョン技術に置き代わることとは予測される。送信及び記憶の両方においてアナログからデジタル画像への転換は、アナログオーディオレコードから現在どこにでもあるコンパクトディスクに切り換えられたことと同様である。

【0067】デジタル画像技術が一般的に有用となるためにデジタル画像を符号化する標準案が採用されている。そのような標準案はJPEG規格として知られ、静止画用として用いられる。動画用としては2つの標準、MPEGとH.261とが存在あり、その両方は動画の連続するフレームの各々においてJPEGと同様の処理を行なう。JPEGを繰り返し用いること以上の利点を得るために、MPEG及びH.261は連続するフレーム間の差において動作し、その差、すなわちフレーム間の動きが小さいという良く知られたことを利用する。よって、一連の画面の中の各フレームが一連の画面の中の最も近いフレームに完全には似ていないような同等の静止画情報を送信又は記憶することよりむしろその変化に対応する情報を送信又は記憶することは時間又は容量を少なくする。

【0068】便宜のために全ての現在の標準は画像又は絵をタイル又はブロックに分割している。各ブロックは幅8ピクセルで高さ8ピクセルの画像片からなる。各ピクセルはピクセルの「成分」として知られた3つ(又はそれ以上)のデジタル数によって表される。カラー化されたピクセルを例えば、YUV, YCr, Vb, RGB等の標準表記を用いて成分に分割する異なる方法は沢山ある。従来のJPEGのような方法は各成分で分かれて作用する。

【0069】目は画像の高い周波数成分(又は縁)に鈍感であることは良く知られている。最も高い周波数に関する情報については画像の重大な低下を気付かせることなく普遍全て削除することができる。目が情報の欠落を分ることなく高周波数情報を除去することにより画像の中の情報内容を減少させるためには空間情報(例えば、輝度の実値)を含む8×8のピクセルブロックは周波数情報を得るために何らかの方法で変換されなければならない。JPEG、MPEG及びH.261の標準規格は8×8周波数マトリックスを得るように8×8空間マトリックスについて動作する公知の離散コサイン変換(Discrete Cosine Transform)を使用している。

【0070】上記したように、入力データは画像の正方形のエリアを表している。入力データを周波数表裏に変換する場合に、適用される変換は2次元でなければならないが、そのような2次元変換を効率よく計算すること

は難しい。しかしながら、その公知の離散コサイン変換 (DCT) 及びそれに関連した逆DCT (IDCT) は分離できるという特性を有している。これは一度に8×8ピクセルブロックの64ピクセル全てに作用する必要があることよりむしろ、そのブロックを先ず、列毎に中間値に変換することができることを意味する。その中間値は最終的な変換した周波数値にコラム毎に変換される。

【0071】N次 (order N) の1次元のDCTは2つのN×Nのマトリックスを乗算することと同等である。8×8のピクセルブロックについて必要なマトリックス乗算を行なうためには、512回の乗算及び448回の加算が要求され、それによって1024回の乗算及び896回の加算が8×8のピクセルブロックについて完全な2次元のDCTを行なうために必要である。それらの演算動作、特に乗算は複雑で遅く、達成できる送信レートを制限する。それらはDCTを実行するように用いられるシリコンチップにかなりのスペースを必要とする。

【0072】DCT処理では要求される計算量を減少させるように再構成することができる。現在、DCTで必要とされる計算を減少させる方法が2つあり、その両方とも「2進間引き (binary decimation)」を用いている。その「2進間引き」手段はN×Nの変換よりも2つのN/2×N/2の変換を用いることにより計算することができ、これを準備する間に計算オーバーヘッドを加える。8×8変換は512回の乗算及び448回の加算を必要とするが、4×4変換は64回の乗算及び48回の加算を必要とするだけである。2進間引きは284回の乗算及び352回の加算を節約し、その間引きを行なう場合に必要なおバヘッドは計算における減少に比較して取るに足りない。

【0073】現在、2進間引きの2つの主な方法がEong Gi Lee (「A New Algorithm to Compute the DCT: DCT計算の新たなアルゴリズム」IEEE Transaction on Acoustics, Speech and Signal Processing, Vol. Aspp 32, No. 6, p 1243 December 1984) 及びWen-Hsiung Chen (「A Fast Computational Algorithm for the DCT: DCT用の高速計算アルゴリズム」Wen-Hsiung Chen, C. Harrison Smith, SCPralick, IEEE Transaction on Communications, Col. Com 25, No. 9 1004, September 1977) によって開発された。前者 (Eong Gi Lee) の方法は逆DCT

致を用いて再帰的 (recursive) 2進間引き方法を定める。この前者のアプローチはIDCTに適切にだけである。

【0074】後者 (Wen-Hsiung Chen) の方法はマトリックスを対角線だけに減少させる再帰的マトリックス一致を使用する。この方法は対角線マトリックスのための知られた一致を用いてDCTの容易な2進間引きを与える。両者 (Lee及びChen) の方法の大きな欠点は、乗算及び加算を行なう必要があるときにに関して不平衡なことである。主に、それらの方法の両方共に多くの加算に続いて多くの乗算、又はその逆に多くの乗算に続いて多くの加算を要求する。Lee又はChenの方法をハードウェアに行なうときには加算器及び乗算器のバラバラ動作を備えることができない。これは、ハードウェアの最善の活用は全ての加算器及び乗算器が常に使用されるときであるのでそれらの速度及び効率を低下させる。

【0075】そのようなDCT及びIDCT動作を行なう公知の方法及び装置の更なる欠点は、いわゆる正規化係数を扱うことが通常難しく、公知のアーキテクチャは全ての乗算器が使用されているとき追加の乗算時間を加える必要があることである。順及び逆のDCTをビデオデータに適用する公知の方法は、計算を行なう半導体素子のレイアウトと関係させる必要がないソフトウェア設計者にとって非常に簡単で高効率である。しかしながら、そのような方法はデジタルビデオ用で望まれる送信レートで満足に行なうためには遅過ぎ、又は半導体アーキテクチャ及びハードウェア相互接続において複雑すぎる。

【0076】ビデオデータによりDCT及びIDCT動作を行なう存在する方法及びハードウェア構成の他の欠点は、それらが数値の浮動小数点内部表示を要求することである。この欠点を表すため、(たとえあるとしても) 10進小数点の右に数字を含む3桁の数で処理することかできるだけの計算機を有すると仮定する。更に、その計算機は数12.3と4.56とを加えることであると仮定する (10進小数点はそれら2つの数値のおける数字の位置に関連して固定されていない。言い換えれば、10進小数点は「浮動」であることができる。) 計算機は数16.86を表すために要求される4つの数字を記憶することができないので、計算機はその答を3つの数字に減少させるなければならない。最も右の「6」を省略することによりその答を切り詰めて16.8の答を生じさせるか、3桁の近似値16.9までその答を四捨五入する必要なハードウェアを有していなければならない。

【0077】非常に簡単な例を示すと、浮動小数点演算が要求されるならば、正確でなくなることを受け入れるか、四捨五入誤差を最小にするため非常に複雑で空間を無駄にする回路を含まなければならない。しかしながら

ら、効率の良い四捨五入回路でさえ、四捨五入又は切り捨て誤差の累算及び伝播はビデオ信号において受け入れることができない歪みを導く。この問題は、浮動小数点四捨五入又は切り捨て誤差が加算より乗算で一般的に大きいのでビデオ信号を処理する方法が多数の乗算を要求するとき更に大きい。

【0078】より効率的なDCT/IDCT方法及びハードウェア構成は、その方法で使用された数値が固定の10進小数点で、各数値のフルダイナミックレンジを使用できるような方法で表わすことができる。そのようなシステムにおいては、切り捨て又は四捨五入誤差が除去されるか、少なくとも大きく減少される。上記の例では、ハードウェアが4つの数字を扱うことができるならば、99.99より大きい数は必要とされず、どの数も第2及び第3位置の間に10進小数点を有し、それでの10進小数点の存在は計算に全く影響を与えない。よって、どの数も整数であるかのように演算を実行することができる。例えば、その答1230+0456=1686は、「1686」の中の「16」と「8」との間に10進小数点を有するべきあることは常に分かるので、12.30+4.56=16.86と同等であることは明らかである。その代わりに、数値(定数又はそれ以外)が同一の範囲内に存在するように選択的に位置どり又は調整されるならば、その範囲内の各数を1組の整数として正確にかつ明確に表わすことができる。

【0079】必要とされた乗算器の数を減らす1つの方法は、異なるソースからの入力データを受け入れることができる単一の乗算器を単に備えることである。言い換えれば、あるアーキテクチャはDCT又はIDCT計算の異なるステップにおいて要求される乗算を行なう単一の乗算器を使用する。そのような「クロスバー切替」は要求される乗算器の数を減少させるけれども、大きく複雑な乗算器の構成は乗算器への入力の選択、乗算器からの他の分離、及び選択したソースから乗算器の入力への適切な信号の切替をそれに代わって含まなければならぬことを意味する。また、追加の大規模の乗算器は共有の乗算器から適切な後段の回路に多数の入力を切り換えることが要求される。よって、クロスバー切替又は多重化は複雑であり、(余分の記憶が必要であるので)一般的に遅く、価格は最終の半導体実現においてかかる。

【0080】「クロスバー切替」を含む現存アーキテクチャの他の欠点はそれらが一般用途の乗算器を必要とすることである。言い換えれば、現存のシステムは両方の入力が可変できる乗算器を必要とする。良く知られたように、デジタル乗算器の表現とは、乗算ワードの現ビットが「1」ならば被乗数の値はその部分結果に加算され、現ビットが「0」ならばそうされないように加算器及びシフトの行を一般的に含む。一般用途の乗算器はどのビットも「1」である場合を処理することができなければならぬので、加算器の列は乗算ワードの各ビット

に備えられる必要がある。

【0081】例として、データワードが8ビット幅で、5による単一入力を乗算すると仮定する。数5の8ビット表現は0000101である。言い換えれば、5によるデジタル乗算は入力値が左に2位置シフトさせ(4による乗算に対応する)、そしてそのアップシフトされた値に加算されるような入力値を必要とする。係数の他の6つの位置は「0」のビット値を有し、それではシフト又は追加のステップを必要としない。

【0082】固定係数の乗算器、すなわちこの場合、5だけによる乗算をすることができる乗算器は(キャリビットを扱うために必要な回路に無関係なく)乗算を行なうためには単一のシフト及び単一の加算器だけを必要とする。逆に、一般用途の乗算器は8つの位置の6つは決して使用の必要はないが、8つの位置の各々にシフト及び加算器を必要とする。例示のように、係数内の各「0」に対応する加算器の行を設計者が除去することができるので、固定の係数は乗算器を簡略化することができる、それによりシリコン領域を少なくすることができる。

【0083】IDCT方法において、本発明によれば、N×N画素ブロックの各N行(row)及びN列(column)毎の1次元IDCTは間引きされ、1次元IDCT(1-D IDCT)はN-2偶数の画素入力ワード及びN-2奇数画素入力ワードで分離して行なわれる。好ましい実施例においては、JPEG規格に準じてN=8である。2次元IDCTの結果は2つの1次元IDCT動作を(データの中間の順序を置き換えて)順に行なうことにより得られる。

【0084】共通の処理ステップにおいて、N=8のためには第1対の入力値は乗算のために加算器及び減算器を出力する必要な通過される。第2対の入力値の各々は2つの基準化コサイン値に対応する2つの一定係数値の各々によって乗算される。他の乗算と1つの減算及び1つの加算だけはその共通の処理ステップでは要求されない。その第2対の入力値又は奇数の結果の値を得るため第1対の入力値と対(pairwise)で加算され又は差し引かれる。

【0085】事前共通処理段では、最も低位の奇数入力ワードは2の平方根によって予め乗算され、その奇数入力ワードは共通の処理ブロックにおける処理の前にバウイズで合計される。事後共通処理段では、処理した奇数入力ワードに対応する中間値は奇数の結果値を生成するために所定の一定係数によって乗算される。奇数及び偶数の結果値の計算後、N/2高位出力は偶数の結果値からの奇数の結果値の簡単な減算によって生成され、N/2低位出力は奇数の結果値と偶数の結果値との簡単な加算によって生成される。

【0086】(ビデオ処理システムの送信端における)DCT及び(本発明の様々な態様の1又はそれ以上を組

み入れている受信端における) I D C T の両方のために、その値は簡単な 2 進右シフトによる 2 の因数によって下方に故意に調整される。この平衡した上方への調整は従来の方法において要求される多数の乗算ステップを除く。

【0087】本発明の方法における他の態様においては、一定係数の選択したビット又は中間の結果データワードは「1」又は「0」のいずれかに選択したビットの所定の設定によって四捨五入又は調整される。画素データの 2 次元変換は第 1 の 1 次元 I D C T 処理ステップから 10 10 の出力値による第 2 の同一の 1 次元動作によって実行される。

【0088】本発明の他の態様によれば、I D C T システムは事前共通処理回路と、共通処理回路とを含んでおり、そこにおいて事前共通、共通及び事後共通処理計算は入力データワードによって行なわれる。管理コントローラは様々なシステムラッチの読み込みを制御するために制御信号を発生する。好ましくは、それは低位出力信号を生成しラッチするための偶数及び奇数の結果値の直接加算及び高位出力信号を生成しラッチするための偶数 20 20 の結果値からの奇数の結果値の直接減算に事前共通ブロックのラッチを入力させるように N/2 偶数及び N/2 奇数入力ワードのアプリケーションを時系列多重化し、内部マルチプレクサを連続的に制御することである。

【0089】本発明においては、偶数及び奇数の入力ワードは好ましくは同一の処理ブロックを介して分離した経路で処理される。入力データワードは好ましくは(必要ではないが)、厳格に昇順又降順ではなく、むしろデータ通路の効率の良い「バタフライ」構成を可能にする 30 30 順でラッチされる。更に、少なくとも共通処理回路はその適当な動作のために要求されるクロックや制御信号を用いることなく、他の処理ブロックであるように特定のアプリケーションに従う事前ロジック回路として構成される。

【0090】一般用途の(2つの可変入力を備えた)乗算器は要求されない。むしろ、一定係数乗算器が好ましい実施例の中では含まれる。更に、固定少数点整数演算\*

\* 素子とその本発明の好ましい実施例では含まれ、次の 1 又はそれ以上の特徴を有するビデオデータの I D C T 変換を行なう方法及びシステムを提供するように設計することができる。

1. 全ての高価な演算動作の連続的使用。
2. I D C T を実現するために必要なシリコン領域を減少するために、好ましくはアーキテクチャの有効なパイプライン処理のために要求されるより多くなく、追加の記憶素子を必要とする一般用途の乗算器よりむしろ少数の一定係数乗算器と接続したラッチのような少数の記憶素子数がある。
3. 例えば、公知の「リップル加算器」を用いるならば各演算動作が複雑な設計を用いる必要がないように動作は取り決められる。遅延を避けつつ大きなスループット及び効率を可能にするために他の考案がリアレンジ動作を先にするような方法に動作が構成されるならば、「分解」又はそれらの回答を生成するために十分の時間を与える。
4. 自然順に結果を生成することができる。
5. 高価で複雑なクロスバ切替が要求されない。
6. アーキテクチャは速い動作をサポートすることができる。
7. 変換ハードウェアを介したデータの流れを制御するために使用される回路を領域内で小さくすることができる。

#### 【0091】発明の理論背景

本発明による I D C T システムに用いられる信号処理方法の様々な構成要素の目的及び機能と共に利点の理解のためにはそのシステムの理論の概観を理解することが手助けとなる。

#### 2 次元 I D C T の分離可能性

N×N 画素ブロックの 2 次元フーワード離散コサイン変換の数学的定義は次の通りである。ここで、U(j, k) は画素絶対値 X(m, n) に対応する画素周波数値である。

【0092】

【式 1】

$$y(j,k) = \frac{2}{N} (j)c(k) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} X(m,n) \cos \left[ \frac{(2m-1)jn}{2N} \right] \cos \left[ \frac{(2n-1)kn}{2N} \right]$$

ここで、j, k=0, 1, ..., N-1

j, k=0 に対して c(k) = 1/√2 それ以外

1

2N は変換の d c レベルを定め、係数 c(j), c(k) は公知の正規化ファクタである。それに対応する 50

逆離散コサイン変換(I D C T)の表現は次の通りである。



【0093】

$$x(m, n) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} c(j) C(k) Y(j, k) \cos \left[ \frac{(2m-1)j\pi}{2N} \right] \cos \left[ \frac{(2n-1)k\pi}{2N} \right] \quad \text{【式2】}$$

ここで、 $j, k=0, \dots, N-1$  $j, k=0$  に対して、 $c(j), c(k)=1/\sqrt{2}$ 

それ以外 1

フォワードDCTは周波数表現に（輝度等の特性を直接表すかMP EG規格内等の差異を表す）空間値を変換するために用いられる。逆DCTはその名称に示されたようにその逆の「方向」の演算動作を行なう。すなわち、DCTは周波数値を空間値に戻す変換を行なう。\*

※【0094】式2では各コサイン関数は合計符号のいずれか1にだけ対応していることを表す。式2は次のように書き換えることができる。

【0095】

【式3】

$$x(m, n) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} c(j) \cos \left[ \frac{(2m-1)j\pi}{2N} \right] c(k) Y(j, k) \cos \left[ \frac{(2n-1)k\pi}{2N} \right]$$

これは第1のIDCT演算の出力を入力として用いて第2の1次元IDCTによるそのままの標準データの置き換えの後に、 $k$ 及び $n$ に対応する全ての項の積を行なう第1の1次元IDCTと等価である。

★1次元のN点のIDCT（ここで $n$ は偶数）は次の式によって定義される。

【0096】

【式4】

1次元IDCTの定義

$$X(k) = \sum_{n=0}^{N-1} c(n) \cdot y(n) \cos \left[ \frac{n(2k-1)\pi}{2N} \right] \quad k = \{0, 1, \dots, N-1\}$$

 $n=0$  に対して、 $c(n) = 1/(\sqrt{2})$ 

それ以外

1

ここで、 $y(n)$ は逆変換回数へのN入力であり、 $x(k)$ はそのN出力である。2次元の場合のように、DCTの式は合計符号の外の正規化定数と式内で位置を切り換える $x$ 、 $y$ ベクトルとを除いて合計符号の元で同一の構造を有する。

1次元IDCTの分解

上記したように、2次元IDCTについては置き換えによって分離された1次元IDCTのシーケンスを用いて計算することができる。一実施例によれば、それら1次元演算の各々は、要求サイズ及び半導体実現の複雑さを更に減少させるように開発されるサブプロシージャに順に分解される。

【0097】係数の正規化

上記したように、IDCTハードウェアの重要な設計目標は、回路内に備える必要のある乗算器の数の減少である。よって、IDCTにおけるDCTを計算するほとんどの方法は必要な乗算数の減少を試みることである。しかしながら、この実施例によれば、全ての入力値は2の平方根のファクタによって強制的に上方調整される。言い換えれば、本発明のこの実施例による方法を用いて

ば、IDCT表現式Eの右手側は2の平方根によって乗算される。

【0098】実施例によれば、2つの1次元IDCT演算動作は最終的な2次元IDCTの結果を生成するように（中間の置き換えと共に）並行して行なわれる。それらの1次元演算動作の各々は同一の2の平方根によって乗算される。中間の置き換えは調整を含んでおらず、並行した2の平方根による2つの乗算の結果は、最終的な2次元結果がファクタ（数値）2によって上方調整されることである。無調整値を得るためには、回路は2による除算だけが必要とする。その値は全て数値として表されるので、これはデータの簡単な右へのシフトによって容易に達成することができる。以下に明確にするように、各1次元IDCT段において2の平方根による上方調整及び最終的な2による下方調整はシステムのハードウェア内の加算器、乗算器及びシフトによって達成され、それによってシステムはそのシステムと接続される他の装置に調整した入力の必要を求めない。この理由のため、そのシステムはJPEG又はMP EG規格に準じた動作を行なう他の従来の装置と互換性がある。よつ

て、本発明のこの実施例による正規化は、少なくとも2つの2乗算演算の平方根に対してIDCT半導体アーキテクチャ内のハードウェア乗算器の必要性を除去する。

【0099】以下に詳細に述べるように、各1次元演算動作で入力データの(2の平方根によって上方調整する)単一の追加の乗算ステップは、従来の方法を用いたとき要求される他の乗算ステップの除去をもたらす。

1次元IDCTの高位及び低位出力への分離

表現式Eは $N/2$ 低位出力( $k=0, 1, \dots, N/2-1$ )及び $N/2$ 高位出力( $k=N/2, k=N/2+1, \dots, N$ )に分離して評価することができる。 $N=8$ については、これは先ず、 $y(0), y(1), y$

\* (2) 及び  $y(3)$  を計算するために入力を変換し、そして  $y(4), y(5), y(6)$  及び  $y(7)$  を計算するために入力を変換することを意味している。

【0100】高位出力( $k=N/2+1, \dots, N$ )のために変数  $k' = (N-1-k)$  を導入してそれにより  $k$  が  $(N/2+1)$  から  $N$  まで変化するように  $k'$  は  $(N/2-1)$  から  $N$  まで変化する。それは表現式Eがの式5 (合計の間隔を除いて式Eと同一式) 及び式6に分割することができることを示している。

低位出力:

【0101】

【式5】

$$x(k) = \sum_{n=0}^{N-1} c(n) \cdot y(n) \cdot \cos \left[ \frac{\pi(2k-1)n}{2N} \right]$$

ここで、 $k=\{0, 1, \dots, (N/2-1)\}$ ; and

$n=0$  に対して、 $c(n) = 1/\sqrt{2}$       それ以外      1

高位出力:

【0102】

※ 【式6】

$$x(k) = x(N-1-k) = \sum_{n=0}^{N-1} y(n) \cdot (-1)^n \cos \left[ \frac{\pi(2k-1)}{2N} \right]$$

ここで、 $k=\{N, \dots, (N/2+1)\} \rightarrow k'=\{0, 1, \dots, (N/2-1)\}$

(全ての高位項で  $c(n)=1$  であるので  $c(n)$  はこの式表現には含まれない。)

式5及び式6共に、項  $(-1)^n$  が上位の  $N/2$  出力値で奇数入力 ( $n$  odd) の合計符号の元で積の符号を変えることを除いて、また  $y$  項が  $c(0) = 1/2^{1/2}$  によって乗算されることを除いて、合計符号の元で同一の構造を有する。

【0103】1次元IDCTの偶数及び奇数出力への分離

★

★ 1次元IDCT表現式4における1つの合計は2つの合計に分離することができる。2つの合計は偶数入力 ( $N=8$  では  $y(0), y(2), y(4)$  及び  $y(6)$ ) に対する合計と、奇数入力 ( $N=8$  では  $y(1), y(3), y(5)$  及び  $y(7)$ ) に対する合計とである。 $g(k)$  は偶数入力の部分合計を表わし、 $h(k)$  は奇数入力の部分合計を表わす。よって、

【0104】

【式7】

$$g(k) = \sum_{n=0}^{N/2-1} c(2n) y(2n) \cos \left[ \frac{\pi(2k-1)2n}{2N} \right] = \sum_{n=0}^{N/2-1} c(2n) y(2n) \cos \left[ \frac{\pi(2k-1)n}{2(N/2)} \right]$$

ここで  $k=\{0, 1, \dots, (N/2-1)\}$ ;

【0105】

【式8】

$$h(k) = \sum_{n=0}^{N/2-1} y(2n+1) \cos\left[\frac{\pi(2k-1)(2n+1)}{2N}\right]$$

ここで  $k=(0,1,\dots,(N/2-1))$ .

N=8では式7及び式8における合計はN={0, 1, 2, 3}に直ってとられる。公知のコサインの同一式を思い起こすと、 $2\cos A \cos B = \cos(A+B) + \cos(A-B)$ であり、 $A=\pi(2k+1)/2$ 、 $B=\pi(2n+1)/2$ と設定\*

$$h(k) = \frac{1}{2\cos\left(\frac{\pi(2k-1)}{2N}\right)} \sum_{n=0}^{N/2-1} [y(2n+1) + y(2n-1)] \cos\left[\frac{\pi(2k-1)n}{2(N/2)}\right]$$

ここで、 $k=(0,1,\dots,(N/2-1))$ .

入力  $[y(2n+1) = y(2n-1)]$  は、 $h(k)$  ※に對にされていることを意味している。N=8に対しての計算において奇数入力項がN/2の對の入力p(n) 20 は、p(n)の値は次のようになる。  
=  $[y(2n+1) = y(2n-1)]$  を形成するため※

n	p(n)
0	y(-1) + Y(1) = Y(1) Y(-1) = 0 定義により
1	y(1) + y(3)
2	y(3) + y(5)
3	y(5) + y(7)

$h(k)$ の式9は次のように表わすことができる。

【0107】

【式10】

$$h(k) = C_k \sum_{n=0}^{N/2-1} p(n) \cos\left[\frac{\pi(2k-1)n}{2(N/2)}\right]$$

ここで、 $k=(0,1,\dots,(N/2-1))$ .

合計符号の元でコサイン項は(式5と比べて)1次元IDCTの構造を各々有するg(k)及びh(k)の両方に対して同一であることに注目されたい。しかしながら、奇数項すなわちh(k)に対するIDCTの結果はファクタ  
 $C_k = 1/\{2\cos[\pi(2k+1)/2N]\}$ によって乗算される。

【0108】言い換えれば、g(k)は偶数入力y(2n)で演算動作するn/2-ポイントIDCTであり、

\*すると、

$$2\cos A = 1/\{2\cos[\pi(2k+1)/2N]\} = C_k$$

によって式8の両側を乗算することができる。C<sub>k</sub>は合計のインデックス(指標)nとは関連がないので、合計符号内で動かすことができる。y(-1)=0という定義を仮定し、入力y(7)に対するコサイン関数が0に等しい。h(k)は次の式に書き換えることができる。

【0106】

【式9】

h(k)は  $[y(2n+1) = y(2n-1)]$  で演算動作するn/2-ポイントIDCTであり、ここで、定義によってy(-1)=0である。次に同一を示す。

$$\begin{aligned} 30 \quad & y_n = y(n); \\ & c_1 = \cos(\pi/8); \\ & c_2 = \cos(2\pi/8) = \cos(\pi/4) = 1/\sqrt{2}; \\ & c_3 = \cos(3\pi/8); \\ & d_1 = 1/[2\cos(\pi/16)]; \\ & d_3 = 1/[2\cos(3\pi/16)]; \\ & d_5 = 1/[2\cos(5\pi/16)]; \\ & d_7 = 1/[2\cos(7\pi/16)]; \end{aligned}$$

更に、調整したコサイン係数は次の如くある。

$$\begin{aligned} & c_1 s = (\sqrt{2}) \cos(\pi/8); \\ & c_3 s = (\sqrt{2}) \cos(3\pi/8); \\ & \text{コサイン関数の均一性 } (\cos(-\phi) = \cos(\phi)) \text{ 及び周期性 } (\cos(-\phi)) \pi(-\phi) = -\cos(\phi) \text{ を用いて、式7及び式8は} N=8 \text{ に対して} \\ & \text{拡張することができる。} (0) \text{ は } 1/\sqrt{2} \text{ である。} \end{aligned}$$

44

及び

【0 1 1 1】

\* 【式12】

$$c_2 = \cos(\pi/4) = 1/\sqrt{2}$$
であるので、 $\sqrt{2}$ に 30 【式14】

$$\begin{array}{|c|} \hline g(0) \\ \hline f(1) \\ \hline g(2) \\ \hline g(3) \\ \hline \end{array} = \varepsilon \cdot \begin{array}{|c|} \hline y_0 \\ \hline y_2 \\ \hline y_4 \\ \hline y_6 \\ \hline \end{array}$$

【0 1 1 5】

【式15】

$$\begin{bmatrix} h(0) \\ h(1) \\ h(2) \\ h(3) \end{bmatrix} = D \cdot C \cdot \begin{bmatrix} \sqrt{2} \cdot y1 \\ y1 \cdot y3 \\ y3 \cdot y5 \\ y5 \cdot y7 \end{bmatrix}$$

ここで、 $D = d \text{ i a g } [d_1, d_3, d_5, d_7]$  = 対角線に沿った  $d_1, d_3, d_5$  及び  $d_7$  を備えかつ他のエレメントは 0 に等しい  $4 \times 4$  のマトリックスである。式 14 及び式 15 に示すように、 $g(k)$  を得るために偶数入力で動作するプロシージャと  $h(k)$  を得るために奇数入力で動作するプロシージャとはコサイン係数マ

$$Q = \begin{bmatrix} 1 & c1s & 1 & c3s \\ 1 & c3s & -1 & -c1s \\ 1 & -c3s & -1 & c1s \\ 1 & -c1s & 1 & -c3s \end{bmatrix}$$

50 に奇数入力で動作するプロシージャとはコサイン係数マ

トリックスCによる乗算の共通ステップを有する。しかしながら、 $h(k)$ を得るために入力には先ず(定義による $y = (-1)$ を呼び出して)対で合計しなければならない。 $y = (1)$ は2によって予め乗算されなければならない。Cによる乗算の結果はDにより乗算される必要がある。

【0116】また、上記したように、そのN-ポイント、1次元IDCT(式4を参照)は(グループ化された) $N/2$ 奇数及び $N/2$ 偶数入力値について(合計符号の元で)共通のコア動作を各々含む2つの $N/2$ -ポイント、1次元IDCTに分離することができる。上記の表現はこの実施例において実施されるようにIDCT

つぎの簡単な構造を生成する。

低位出力( $N=8$ , 出力 $k \{0, 1, 2, 3\}$ ):

【0117】

【式16】

$$y(k) = g(k) + h(k)$$

高位出力( $N=8$ , 出力 $k \{4, 5, 6, 7\}$ ):

【0118】

【式17】

$$y(k) = y(N-1-k') = g(k') - h(k')$$

$g(k)$ は出力値を直接生成するため偶数入力値で直接演算動作し、 $h(k')$ は値 $d1, d3, d5$ 及び $d7$ による乗算と同様に入力値をグループ化して含んでいる。通常のように、IDCT回路の設計者はサイズ対速度、実現した素子数の多寡に減少した相互接続の複雑さ等のトレードオフの数に直面する。例えば、シリコンチップ上に追加の又は更なる複雑な素子を含むことによる計算速度を改善することができるが、これは明らかにその具体化を大きく又は更に複雑にしよう。また、IDCTチップで有用である又は望まれることが「ルックアヘッド(先取り)加算器等」の精密で複雑な設計の使用を限定又は阻むことになる。

【0119】正しさの標準

全ての計算の限らない精密性及び正しさ、その上、無限の記憶容量及び計算時間を仮定すると、IDCT及びDCTの変換映像データによって生成される映像は完全に元の映像を再生する。勿論、そのような完全は存在する技術を用いて得られない。

【0120】しかしながら、標準化を達成するために、IDCTシステムは国際電信電話諮問委員会(CITT)によって「CITT勧告H.261-逆変換正確仕様の添付1」として出された標準化方法に応じて現在測定される。このテストは生成されるランダム整数を含む10000組の $8 \times 8$ のブロックを詳述する。それらのブロックは10000組の $8 \times 8$ の「基準」IDCT出力データを生成するため予め設定された正確度を用いて(予め設定された四捨五入、クッビング及び演算動作の先に又はそれに続いて)DCT及びIDCT変換される。

【0121】IDCTの実行が試験されるとき、そのCITTのテストブロックが入力として用いられる。実際にIDCT変換した出力は公知の「基準」IDCT出力データと統計的に比較される。最大値は全体として及び各成分としてブロックのピーク、平均、2乗平均及び平均値の平均誤差の点からそのIDCTに対して特定される。更に、そのIDCTは対応する入力ブロックが全て0を含んでいるならば、全て0出力を生成しなければならない。また、そのIDCTは全ての入力データの符号が変化されたときには同一の標準と合わなければならない。IDCTの実行は、最大の誤差がテスト中に特定した最大値を越えないならば、受け入れられるべき正確性を有することである。

【0122】他の公知の標準は(「 $8 \times 8$ の離散コサイン変換の実行のためのIEEE標準仕様案」, P1180/D2, 1990年7月18日)及び(「 $8 \times 8$ の離散コサイン変換」, ISO委員会のCD1172-2案の添付A)における米国電気電子学会(IEEE)のものである。それらは上記のCITTの標準に本質的に一致している。

【0123】ハードウェアの実現

図9は本発明の一実施例によるIDCT方法のデータの流れを示す簡略ブロック図である(図示及び以下に説明するように、ハードウェアの構成は更に小型で効率的に仕得る)。図9において、 $Y[0]$ 及び $Y[4]$ 等のシステムへの入力及び $X[3]$ 及び $X[6]$ 等のシステムからの出力は単線で伝達されるように示されている。図9の単線の各々は各入力及び出力に対応する完全に並行に多くのビットで幅の広いデータワードを伝送するデータバスの形で多くの伝導体を示していることと理解すべきである。

【0124】図9において、大きな円225及び226は2入力加算器を示し、その加算器の入力の接続点の小さい円227は対応する入力ワードの補数が用いられることを示している。そのような補数をとる入力を用意した加算器は、非補数の入力から補数をとった入力を減算する。例えば、上方左の加算器からの出力 $T0$ は $Y[0] + Y[4]$ に等しくなるけれども、出力 $T1$ の加算器は値 $Y + (-1)$ ,  $Y4 = Y0 - Y4$ を形成する。よって、1つ補数をとる入力を用意した加算器は差をとる成分であるということが出来る。

【0125】また、図9において、一定係数乗算はデータ通路に塗りつぶしの三角形230によって示されている。例えば、入力 $Y1$ は $B0$ を生成する加算器に入力する前に、2の平方根の乗算器を通過する。よって、中間値 $T3 = Y2$ ,  $T3 = Y2 \cdot c1s + Y6 \cdot c3s$ 及び中間値 $B2 = p1 \cdot c3s - p3 \cdot c1s = (Y1 + Y3) \cdot c3s - (Y5 + Y7) \cdot c1s$ である。その示した加算、減算及び乗算を行なうことにより示した構成は $g(0) \sim g(3)$ のために式11及び $h(0) \sim h$

(3) のために式 12 を実行することが分かる。

【0126】図 9 は本発明の重要な利点を示している。図 9 に示すように、その構成は 4 つの主要な領域、すなわち、対となった入力  $p(k)$  を生成した 2 の平方根によって入力  $y(1)$  を乗算する事前共通ブロック PREC (231)、式 12 のように一定係数  $d1, d3, d5, d7$  の 4 つ乗算器を含む第 1 事後共通ブロック POSTC1 (233)、式 16 及び式 17 のように低位出力用として  $g0 \sim g3$  項と  $h0 \sim h3$  項とを合計し、高位出力用として  $g0 \sim g3$  項と  $h0 \sim h3$  項との差を生成する第 2 事後共通ブロック POSTC2 (235)、及び奇数及び偶数データ通路に含まれる共通ブロック CLK (232) に分割されている。本発明の実施例による処理回路においては、奇数及び偶数入力に行なう共通動作は図 9 に示したような複成よりもむしろ単一の構成によって実行される。

【0127】実施例における動作方法及びデジタル構成を理解するためには、「キャリビット」を理解することが手助けとなる。簡単な例として、 $1+1=0$  のような 2 つの 2 進数の加算においては「1」のキャリは正しい結果「10」（十進数「2」の 2 進数表示）を生成するため次の高位ビットに加算されなければならない。換言すれば、 $0+0+1=0$ （キャリなしの合計） $+10$ （キャリワード）である。「合計」に「キャリワード」を加えて正しい回答  $0+0+1=10$  を得る。

【0128】10 進数の例として、数「436」と「825」とを加算すると仮定する。手で 2 つの数を加算する共通の手順は通常、次に示す如くである。

1. 1 の位：「6」と「5」との加算は十の位に「1」のキャリを有する「1」である。

合計：1、キャリイン：0、キャリアウト：1

2. 十の位：「3」と「2」との加算は「5」であり、前ステップからのキャリ「1」を加算してキャリなく「6」を与える。

【0129】

合計：5、キャリイン：1、キャリアウト：0

3. 百の位：「4」と「8」との加算は千の位に「1」のキャリを有する「2」であり、前ステップからのキャリはない。

合計：2、キャリイン：0、キャリアウト：1

4. 千の位：「0」と「0」との加算に百の位からのキャリ「1」を加算して「1」を与える。

【0130】

合計：0、キャリイン：1、キャリアウト：0

回答の「1261」は隣接の低位の位置のキャリアウトによる各位へのキャリインにより同一の位の合計に各位のキャリイン合計を加算して生成される。（最低位へのキャリインは常に 0 である。）勿論、問題は十の位からキャリインがあるか否かを知るまで百の位で「4」と「8」とを加算することを持つ必要があることである。

これは本質的にこのような動作をする「リップル加算器」を表わす。リップル加算器は特別な記憶素子を必要とすることなく「最終の」回答を得るが、他の設計より遅くなる。

【0131】代替の設計として「キャリセーブ (carry-save)」が知られている。各位の 2 つの数の合計は部分合計又は結果ワード（この例では 0251）及び異なるワード（1010）でキャリ値を記憶することにより生成される。その完全な回答は次のステップでその合計及びキャリワードを「分解する (resolve)」ことによって  $0251+1010=1261$  のように得られる。記憶されている限りの時間にキャリワードを部分結果に加算することができるか否かを判別するために待つことなく同時に位毎に加算を行なうことができる。

【0132】その解答動作は各計算段で要求される時間の大きな割合を通常必要とするので、それら動作の速度アップはその変換サイズにおける比較的小さな増加だけが必要とするが、全体の動作速度に重要な影響を与える。よって、キャリセーブ乗算器は各行にリップル加算器を用いることよりも通常速い。しかしながら、乗算器において各加算のためのキャリワードは記憶される次の加算へ渡す必要があるため時間のかかる利点は大きな複雑さを犠牲にしている。更に、乗算の最終出力を得るために、最終部分合計及び最終キャリワードはリップル加算器に加えることにより求めなければならない。しかしながら、1 つのリップル加算器だけが必要となり、それにより時間節約は実行される必要がある乗算の大きさに通常、比例する。更に、キャリワードは加算される他の数として扱われ、最終の乗算解答が必要となる前にときにそれが加算される限りは実際の加算を遅延させることができる。

【0133】本発明のこの実施例において、遅延解答の実現は設計を簡単にし、IDCT 回路の処理能力を増加させるために用いられる。また、予め選択したキャリワードのあるビットは、標準テストデータセットによる本発明のテスト動作の統計的分析に基づき IDCT の結果で大きな期待で正確性を与えるために解答の前に所定値に任意的に及び強制的に変えられる。

【0134】図 10 は本発明の好ましい実施例の構成を示すブロック図である。この好ましい実施例において、奇数及び偶数の入力は時間多重化され、共通ブロック CLK (232) において分離処理される。この入力はどちらか一方の順に処理される。図 10 において、記号  $Y[1, 0]$ 、 $Y[5, 4]$ 、 $Y[3, 2]$  及び  $Y[7, 6]$  は奇数入力  $Y1, Y3, Y5, Y7$  が先ず計算回路を完全に通過し、偶数入力  $Y0, Y2, Y4, Y6$  が続くことを示すために用いられる。この順は現実実施例にかくことができないものではなく、それでもなお、以下に説明するように、下流 (downstream)

への演算動作は、奇数入力だけについて実行され、奇数入力値を先ず入力することによりそれら下流への演算動作を、全ての入力に共通の演算動作が偶数入力について上流(upstream)へ実行されると同時に行なうことができる。これは多数の演算素子が逆にアイドルのままである時間を減少させる。

【0135】同様に、記号X[0, 7], X[1, 6], X[3, 4]及びX[2, 5]は低位出力X0, X1, X2, X3が先ず出力され、それに高位出力X4, X5, X6, X7が続くことを示すために用いられる。図9及び図10に示すように、奇数入力Y1, Y3, Y5, Y7であるため必要はないが、入力は完全に初期的には昇順にグループ化される。この順に入力信号を配置することは図9及び図10に示した簡単な「パタライ」データ通路構成を可能にし、シリコン半導体素子において本発明の実行における相互接続効率を大きく増加させる。

【0136】図10に示したように、加算器及び減算器は「+」（加算器）235、1つの補数入力値を有する加算器「-」（減算器）236又は「±」（加算及び減算237の間を切り換えることができる分解加算器/減算器）のいずれかを囲む円によって示される。2つのmビットの入力ワードの共通ブロック232における大部分の加算器及び乗算器は、加算/減算のキャリビットを含むmビット又はm-1ビットと並列して結果として生じるmビットの部分である。言い換えれば、共通ブロックCBLK 232における第1の乗算及び減算は、キャリビットの加算が次の処理段まで遅延されるという未分解の意味である。このステップの利点はそのようなキャリセーブ加算器/減算器がその結果にキャリビットワードの最終の加算を行なう必要がないということである。しかしながら、分解加算器はまた加算器の出力におけるバス幅を減少させるために用いられる。

【0137】図10は本発明の好ましい実施例における1及び2入力のラッチの使用を示している。図10において、ラッチは長方形238として示されており、事前共通ブロックPREC 231及び事後共通ブロックPOSTC 233の両方に用いられる。1入力のラッチはラッチg[0, 7], g[1, 6], g[3, 4]及びg[2, 5]とh[0, 7], h[1, 6], h[3, 4]及びh[2, 5]からの各出力に対応する計算したg(k)及びh(k)である分解加算器/減算器への入力をラッチすると共に、乗算器D1, D3, D5及びD7の入力で使用される。そのように、分解加算器/乗算器は上記の式16及び式17に示した加算又は減算を行なう。

【0138】上記したように、偶数入力Y0, Y2, Y4及びY6は共通ブロックCBLK 232において処理される前に対にされる必要はない。しかしながら、奇数入力Y1, Y3, Y5, Y7は適切に入力値が共通ブロックCBLK 232に送られることを確かなるために2の平方根によって乗算されなければならない。事前共通ブロックPREC 231は各入力値に2入力乗算(「MUX」)ラッチC10, C54, C32及びC76を有する。2入力MUXラッチへの1入力はその結果として未処理の入力値に直接結合され、他の入力は分解加算器及び入力Y1用の2の平方根の分解乗算器から入力される。よって、正しい対の又は対でない入力を、それら2つの入力の多重化ラッチの単なる切り換えによって共通ブロックCBLK 232に容易に送ることができる。

【0139】図10に示すように、2の平方根の乗算器D1, D3, D5, D7は好ましくはそれらの出力を定める。すなわち、完全な合計を生成するためにキャリビットが加算されている結果を生成する。これはその乗算器からの出力が対応するパラレルデータ通路において未乗算の入力として同一のバス幅を有することを確かなる。

【0140】本発明による共通ブロックCBLK 232の好ましい実施例は、Y[1, 0]及びY[5, 4]各々について的前方データ通路に「ダミー」減算器240を含む。それら素子は2つの入力をパラレル出力として通過するように結合するために動作する(ダミー減算器の場合、1つの入力の2の補数をとった後)。各場合において、1つの入力はそれが次の処理段で加算されるキャリビットを含むように処理される。よって、対応する加算及び乗算は遅延されるけれども行なわれる。

【0141】この技術はフルスケールの加算器/乗算器をそれら素子の実現では必要としないので上方の2つのデータ通路に必要なリソースを減少させる。よって、その「結合器(combiners)」は加算器及び乗算器として動作し、それら素子のために実行させることができ、次の加算のための素子への単なる導体として又は減算のためのインバータ列として実行させることができ、それらのいずれも少しもまたは全く追加回路を必要としない。

【0142】そのような結合器の使用は、共通ブロックCBLK 232における始めの加算器及び減算器からの出力が同一の幅を全て有し、底部の2つのデータ通路にあるキャリセーブ加算器/減算器の出力と一致すること意味している。それらはその2つのデータ通路で共通ブロックCBLK内のそれに続く分解加算器及び減算器への入力を生成する。

【0143】上記したように、偶数入力Y0, Y2, Y4, Y6は共通ブロックCBLK 232において適切に入力値が共通ブロックCBLK 232に送られることを確かなるために2の平方根によって乗算されなければならない。事前共通ブロックPREC 231は供給し、対の値p0~p3(図9を参照、p(n)の定義)を記憶する多重ラッチC10, C54, C32, C76の低位入力(図10を参照)を選択す

る。ラッチ1h0、1h1、1h3及び1h2は各々値H0、H1、H3及びH2をラッチするために活性化される。

【0144】その管理制御回路は事前共通ブロックPREC 231内の2入力多重ラッチC10、C54、C32、C76の上側入力をラッチし選択し、偶数入力ワードをそれらラッチに供給する。偶数入力はg0~g3の値を生成するために用いられるので、管理制御回路は事後共通ブロックPOSTC 233内のラッチLg0~Lg3をg(k)値を記憶するために制御する。

【0145】g(k)及びh(k)値がラッチされると、事後共通ブロックPOSTC 233は分解加算器/減算器を減算モードに切り換えることにより高位信号X7、X6、X5及びX4を出力する。低位出力信号X3、X2、X1及びX0は分解加算器/減算器を加算モードに切り換えることにより発生される。出力データについては自然の順を含む任意の順に送ることができる。

【0146】本発明による好ましい多重化の実行は図10に非常に簡単な図形として示され、図9に示された非多重化構成と同一の計算を行なう。しかしながら、共通ブロックBLK 232における加算器、減算器及び乗算器の数は半分に削減され、ダミー加算器/減算器の使用は高価な演算回路の複雑さを更に減少させる。図11は本発明の実施例によるIDCT回路の実際の主成分及びデータラインを示している。その主成分は事前共通ブロック回路PREC 231、共通ブロック回路BLK 232、及び事後共通ブロック回路POSTC 233を含む。また、そのシステムは入力、タイミング、及び制御信号を事前共通ブロック回路PREC 231及び事後共通ブロック回路POSTC 233に直接又は非直接に供給するコントローラCNTL 241を含む。

【0147】本発明の好ましい実施例においては、入力及び出力信号(Y0~Y7及びX0~X7)は2ビット幅である。これが存在する工業標準によって測定できるように許容できる正確性を確保することができるように、この最小幅は「1」又は「0」であるべき選択された演算素子内のあるキャリアワードを強制することにより部分的に得られる。本発明によれば、あるデータワードの調整に相当するこのビット操作は公知の入力テストデータのIDCT変換を用いた後に、IDCTシステムの結果の統計的な分析の結果として行なわれる。あるビットを所定値に強制することによって、四捨五入及び切り捨て誤差の影響を減少させることができることが分かり、それによってIDCTシステムからの空間出力データが公知の「正しい」空間データから少ない差異にすることができる。しかしながら、本発明の実施例による回路に用いられる成分は公知の方法を用いて異なるパス幅に全て適応させることができるので、本発明は他

のデータワード長に同様に適用することができる。

【0148】共に処理される全ての4つの入力は88の平行導体(4×22)を介して事前共通ブロックPRECに同期して入力されるが、画素ワードは送信データからの時点に1に通常変換される。実施例によれば、入力データワードは好ましくは1つの22ビット入力バスを介して直列に全て搬送され、各入力ワードはデータ通路内の適切な入力ポイントで順にラッチされる。図11に示したように、22ビットの入力データバスはT<sub>IN</sub>[21:0] 242で示されている。

【0149】その図において、以下に説明するように、多数ビット信号はコロン「:」の左の高位ビットとコロンの右の最下位(LSB)を有するブラケットに示される。例えば、入力信号T<sub>IN</sub>[21:0] 242は0から21までの番号のビットを有する22ビット幅である。単一のビットは平方ブラケット内の単一番号として定義され、T<sub>IN</sub>[1]は信号T<sub>IN</sub>の最下位ビットの次のビットを示す。

【0150】本発明の実施例においては、次の制御信号は事前共通ブロックPREC 231の動作を制御するために用いられる。

IN\_CLK, OUT\_CLK: 本発明によれば、そのシステムは好ましくは重なり合うことがない2つの相クロックを使用する。その信号IN\_CLK及びOUT\_CLKは入力、中間及び出力信号の値を保持するラッチの列である。

【0151】LATCH10, LATCH54, LATCH32, LATCH76: 好ましくは、1つの22ビットワードは1度にシステムに入力される。一方、4つの入力信号は一度に処理される。各入力信号は3つの他の入力ワードと処理される前にそのアーキテクチャ内で適切な位置でラッチされなければならない。それらのラッチ信号は各入力ラッチを動作させるために用いられる。例えば、信号LATCH54は先ず入力信号Y5をラッチし、その後、次の処理段階の間だけ入力信号Y5(図10を参照)と同一点で事前共通ブロックPREC 231に入力する入力信号Y4をラッチするために用いられる。

【0152】LATCH: 4つの偶数又は奇数入力信号が事前共通ブロックPREC 231にラッチされると、好ましくはラッチのその後列へ同じ時点でシフトされる。信号LATCHは事前共通ブロックPREC 231内の演算素子によって演算動作される4つの入力値を保持する入力ラッチの第2列を動作させるために用いられる。

【0153】SEL\_BYP, SEL\_P: 図10に示されるように、ラッチC10、C54、C32及びC76にラッチされる偶数入力信号は加算器及び2の平方根の分解乗算器を迂回するようにされるべきである。しかしながら、奇数入力信号は対の入力p(n)を形成する



ために先ず対にされなければならない。信号 Y1 は 2 の平方根によって乗算される必要がある。制御信号 SEL\_P はその対にされた入力信号を選択するために作用される。従って、それらの信号は事前共通ブロック PREC 231 の出力ラッチへ制御信号を通過させるマルチプレクサとして動作するゲートを制御するために用いられる。

【0154】上記したように、厳格な昇順に入力信号を配置しないことは高相互接続効率を有する簡単な「バタフライ」バス構造をもたらす。また、述べたように、奇数入力が入る、事前共通ブロックヘグループとして供給され、それに続いて偶数入力となる。しかしながら、順番は各奇数又は偶数グループ内で用いられれば良い。すなわち、入力の順は奇数入力へ処理するために分離して与えられる、又は回路の分離領域内に少なくとも与えられる如く適切なラッチ配置を用いることができる。

【0155】管理制御回路は事後共通ブロック POSTC 233 のためにタイミング及び制御信号を発生する。それらの制御信号は次の如くである。

EN\_BH, EN\_GH: 図 9 を再度参照すると、共通ブロック CBLK 232 からの出力は、奇数入力を処理した後、H0, H1, H3 及び H2 として示される。それらの信号は最初の共通ブロック CBLK 1 232 内の係数乗算器 d1, d3, d7, d5 各々に送出される。信号 EN\_BH は係数乗算器において乗算された後、h0~h3 値を保持するラッチを動作させると共に g0~g3 値を保持するラッチを動作させるために用いられる。

【0156】ADD, SUB: 図 10 に示したように、実施例は低位出力を生成するために合計及び差 g(k) 及び h(k) 値を求める分解加算器/減算器の 1 つのバンクを含む。信号 ADD, SUB はその分解加算器/減算器を加算及び減算モード各々に設定するために用いられる。

EN\_O: この信号は分解加算器/減算器からの結果をラッチする出力ラッチを動作させるために用いられる。

【0157】MUX\_OUT70, MUX\_OUT61, MUX\_OUT43, MUX\_OUT52: 本発明のよれば、システムからの出力データは好ましくは 1 つの 22 ビット出力バスを介して送出され、それによって、1 つの出力値 (X0~X7) だけが一度に送出される。それらの信号は 4 つのラッチされた出力値の最終の出力ラッチにラッチされるべきものを選択するために順に作用される。よって、それらの信号は 4-1 マルチプレクサのための制御信号として動作する。

【0158】T\_OUT [21:0]: これは事後共通ブロック POSTC 233 からの 22 ビットの信号を示す。事前共通ブロック PREC 231 からの出力信号は共通ブロック CBLK

232 への入力信号を生成するためにラッチされる。図 11 に示されたように、事前共通ブロック PREC 231 からの出力信号は 4 つの 22 ビットのデータワード C110 [21:0], C154 [21:0], C132 [21:0], C176 [21:0] として与えられ、それは共通ブロック CBLK 232 への入力信号 IN [0], IN [1], IN [3], IN [2] となる。

【0159】図 11 に示すように、共通ブロック CBLK 232 からの 4 つの 22 ビットの結果は出力信号 OUT0 [21:0], OUT1 [21:0], OUT3 [21:0], OUT2 [21:0] として並列に変換される。それは、C070 [21:1], C061 [21:1], C043 [21:1], C052 [21:1] として事後共通ブロック POSTC 233 の入力信号としてラッチされる。

【0160】共通ブロック CBLK のために要求される制御信号はない。この例の IDCT システムのユニークな構成のため、システム動作の共通ブロックについてはクロック、タイミング又は制御信号を必要とせず純粋な論理動作として動作することができる。これは更に素子の複雑さを減少させる。アプリケーション (特に、全ての必要な演算動作を行うために十分な時間があるもの) によっては、事前共通ブロック PREC 231 及び事後共通ブロック POSTC 233 がクロック、タイミング又は制御信号なしに動作するように構成しても良い。

【0161】図 12 は本発明における事前共通ブロック PREC 231 のブロック図である。この図及びそれに続く図において、記号「S1 [a], S2 [b], ..., SM [Z]」は信号 S1, S2, ..., SM から選択したビット a, b, ..., z が、信号 S1 の選択ビット「a」である最上位ビット (MSB) 及び信号 SM の選択ビット「z」である最下位ビット (LSB) と共に同一のバスを介して並列に転送されることを表している。ここで、S は任意の信号表示、a, b, ..., z は信号のバス幅の範囲内の整数である。その選択ビットは個々のビットである必要はない。むしろ、全体的又は部分的な多数ビットワードは他の単一ビット (bits)、或いは完全な又は部分的な多数ビットワードと共に送出されても良い。図において、シンボル S は対応する信号表示によって置き換えられる。

【0162】例えば、図 12 において、2 の平方根の乗算器は R2MUL の如く示される。この未分解乗算器から出力される「セーブ (save)」又は「未分解合計 (unresolved sum)」は 21 ビットワード M5S [20:0] として示され、同様に、乗算器 R2MUL から出力された「キャリ」は 22 ビットのワード M5C [20:0] として表され、それはキャリセーブ分解加算器 M5A の「b」入力へのバスを介して転送

される(「0」はセーブ出力の最下位21ビットにMSBとして挿入される。しかしながら、これは分解加算器M5Aの「a」入力へ供給される前になされる。これは記号GND M5S [20:0]で図12には示されている。)言い換えれば、加算器M5AへのMSB入力に対応する導体はグラウンドGNDにそれを接続することにより強制的に「0」にされる。

【0163】「0」がなぜ「合計」の22番目のビットとして挿入されるかを理解するために、乗算の部分合計がn場所分の幅であれば、キャリワードは部分合計に関して左へ1場所分だけシフトされる。よって、そのキャリワードは最下位位置内の「0」(単位位置の中にキャリビットを生成するためこの位置の前には何もないので)を有するn+1番目の位置の有効データビットでn+1場所に拡張される。それらの2つのワードが分解2進数加算器への入力として用いられるならば、キャリワードのビット(数字)が部分合計の対応するビットと適切に位置合わせされることを確実にするように注意しなければならない。これは十進小数点(例えば、整数演算における如く)が両ワード内における位置合わせをもって維持されることを確実にする。加算器への入力がn+1ビット幅とすると、他の入力でキャリワードと位置合わせされるn+1ビット入力を与えるために「0」を全てのnビットのポジティブ部分合計ワードの最高位ビットに挿入することができる。

【0164】上記したように、事前共通ブロックPREC 231内で与えられた時間で処理される4つの入力は入力バスIN (21:0)を介して転送される。この入力バスは4つの入力ラッチIN10L, IN54L, IN34L及びIN76Lの入力に接続される。各ラッチは、入力クロック信号IN\_CLK及び対応するラッチ選択信号LATCH10, LATCH54, LATCH32, LATCH76が高レベル(high)であるときだけ作動可能にされる。よって、ラッチ可能信号LATCH10, LATCH54, LATCH32, LATCH76が引き続いて生じることにより4つの入力をIN\_CLK信号の4つの期間内にそれぞれ各入力ラッチにラッチすることができる。この時間の間に、4つの入力値を安定化しかつラッチするためLATCH信号は入力ラッチIN10L, IN54L, IN34L及びIN76Lを活性化するように低レベル(又は異なる位相)にされるべきである。

【0165】本発明に応じたラッチのタイミングの例は図13に示されている。4つの入力信号が好ましい順にラッチされると、それらはラッチL10L, L54L, L32L, L76Lの第2のバンクへ送られる。それらラッチの第2のバンクは、信号OUT\_CLK及びLATCHが高レベルのときに作動可能にされる。その信号タイミングは図13に示される。

【0166】本発明のシステムは8つの入力ワード全て

の入力を遅延させる必要はない。全ての偶数又は奇数入力ワードが入力ラッチIN10L, IN54L, IN34L及びIN76Lに受け入れられラッチされると、IN\_CLKの次の立ち上がりエッジで遅延なく他の4つの入力信号を受け入れ始めることができるようにその入力ラッチは解放される。

【0167】図の中で示された様々な成分のために用いられた2つの数字のサフィクス記号[10, 54, 32, 76]は、奇数信号が先ず処理され、その構造内のその後のバスにおける偶数信号が続くことを示している。上記したように、この順番は本発明においては要求されず、付加的な順番を用いても良いことはこの分野の当業者には理解され得る。

【0168】4つの入力信号がラッチL10L, L54L, L32L, L76Lの第2のバンクに適切な順番でラッチされると、対応する値は選択バイパス信号SEL\_BYPの活性化により出力ラッチC10L, C54L, C32L, C76Lへの入力として供給されるか、又はそれらは「セレクトp」信号SEL\_Pの活性化により同一の出力ラッチへ対とされかつ乗算された入力として供給される。言い換えれば、全ての信号は事前共通ブロックPREC 231の出力ラッチC10L, C54L, C32L, C76Lへ直接及び非直接的に、演算素子を介して供給される。しかしながら、適切な値は「選択バイパス」信号SEL\_BYP(偶数入力Y0, Y2, Y4及びY6用)又は「セレクトp」信号SEL\_P(奇数入力Y1, Y3, Y5及びY7用)の活性化によりそれらラッチにロードされる。この分野の当業者には理解され得るであろう、それら及びその他の制御信号の所望のタイミング及び順番はコントローラCNT L241の適切な動作構造及び又は「マイクロ」プログラミングによって知知の方法で容易に達成される。

【0169】ラッチL10Lの出力にて最上位の入力値は2の平方根の乗算器R2MULへ先ず供給され、そして示されたように分解加算器M5Aに供給される。分解加算器M5Aからの出力は2の平方根によってラッチL10Lからの出力の分解乗算と等価として示される。その他の3つのラッチL54L, L32L, L76Lからの出力は、22ビットのランチバスLCH54 [21:0], LCH32 [21:0], LCH76 [21:0]を介して直接的に及び分解加算器P2A, P1A及びP3A各々を介して非直接的に、対応する出力ラッチC54L, C32L及びC76L各々に転送される。

【0170】本発明においては、各分解加算器P2A, P1A, P3Aは2つの入力「a」及び「b」を有している。加算器P2Aについては一方の入力はラッチL32Lから入力され、他の入力はラッチL54Lから入力される。(L54Lにラッチされた)入力値Y5及び(L32Lにラッチされた)Y3については加算器P2Aからの出力がY5+Y3に等しく、それは上記したよ

うに p (2) に等しい。従って、その加算器は対の入力値 p (1)、p (2) 及び p (3) を生成するために奇数入力を対にする。勿論、L54L、L32L、L76L にラッチされた偶数入力信号は分解加算器 P2A、P1A、P3A 各々を介して通過するが、その結果として生じる p「値」は出力ラッチ C54L、C32L、C76L へは供給されない。なぜなら、その「選択 p」信号 SEL\_P は偶数入力に対しては活性化されないからである。

【0171】入力クロック信号 IN\_CLK の活性化により出力ラッチ C10L、C54L、C32L、C76L にラッチされる値は、偶数入力 Y0、Y2、Y4、Y6 又は奇数入力について対となった入力値 P0、P1、P2、P3 に等しくなる。入力 Y (1) は値 U (-1) と「対」にされ、それは 0 になると仮定される。図 12 に示されたように、この仮定は値 Y1 に何も加算しないことによって実行される。代わりの、Y1 は図 9 及び図 10 に示したように 2 の平方根によって乗算されるだけである。

【0172】図 14 は本発明における共通ブロック B L K 232 の好ましいアーキテクチャを示している。異なるシステムブロック内の様々な乗算及び加算のための様々な計算を行なう前にその共通ブロックへの入力値を下方調整することが必要又は利点となる。これはシステム内の様々な演算素子への対応する入力のために同一の十進小数点 (それは整数演算のために伴う) の位置を可能にする。

【0173】よって、入力値 IN0 [21:0] 及び IN1 [21:0] はデジタル演算において 2 ビットの右シフトに対応する 4 のファクタによって下方調整される。2 進数表現において数の符号を (正の値を正に、負の値を負に) 維持するために、最上位ビット (MSB) はその結果として生じる右シフトしたワードの 2 つの最上位ビットに複写される。この処理は「符号拡張 (sign extension)」として知られている。従って、入力値 IN0 は符号拡張で 2 ビットだけ下方シフトされて IN [21]、IN0 [21]、IN0 [21:2] として示されたシフトした入力値を形成する。入力値 IN1 [21:0] は同様に 2 の位置だけ符号拡張される。また、入力 IN2 はシフトされ、IN2 [21]、IN2 [21:1] を生成するために拡張される。それらの 1 位置のシフトは 2 のファクタによる切り捨て除算に対応する。

【0174】図 10 に示されたように、入力 IN2、IN3 はその調整された係数 c1s 及び c3s によって乗算される必要があるものである。各入力 IN3 及び IN2 を調整された係数の各々によって乗算される必要がある。図 14 に示すように、これは 4 つの一定係数キャリセーブ乗算器 MUL C1S、MULNC1S、MUL C3C3、及び MULC2S2 によって行なわれる。底

部の IN2 用の乗算器は反転乗算器 MULNC1S である。すなわち、その出力は一定値 C1S により乗算される入力の値の負に相当する。よって、C76 にラッチされた値は (C3S による乗算の後) C32 にラッチされた値から減算される。反転乗算器 MULNC1S を備えることによって、減算は対応する値の負を加えることにより行なわれ、それは差を生成することと同等である。これは後段の加算器に同様の回路の使用を可能にするが、非反転乗算器を使用することはその後段の減算器が利用されることになる。

【0175】本発明の図示した実施例において、4 つのコサイン係数乗算器 MUL C1S、MULNC1S、MULC3C3、及び MULC2S2 が含まれる。もし、構成が信号が乗算器を分離して通過するようにされるならば、必要な乗算は 2 つの乗算器だけ、すなわち c1s 係数用と c3s 係数用を用いて行なうことができる。

【0176】本発明によれば、乗算器 MUL C1S、MULNC1S、MULC3C3、及び MULC2S2 は好ましくはキャリセーブ形であり、それは 2 つの出力ワード、すなわち、ハードウェア乗算器内で行なわれる多数の行の加算の結果に対応する一方と、生成されるキャリビットに対応する他方とをそれら乗算器が発生することを意味している。乗算器からの出力は 2 つの 4 入力分解加算器 BT2、BT3 のいずれかへの入力として接続されている。

【0177】簡単な図示だけのため、乗算器からの 5 つの出力バスは、この分野の当業者には理解できるので、加算器の対応する入力バスに接続されるように描かれておらず省略されている。それらの接続は理解されるべきであり、同一のラベルを有する各出力及び入力によって描かれる。従って、乗算器 MULC1S の同一の出力 M1S [20:0] は加算器 BT3 の入力「sa (save-a)」の低位 21 ビットに接続される。

【0178】図 14 に示されたように、加算器 BT2 及び BT3 への 5 つの入力は「分割 (split)」であるように示される。例えば、加算器 BT2 の「ca」入力是最下位の 21 ビットとしての入力である MC3 [20:0] の上に IN3 [21] を有するよう示されている。同様に、同一の加算器の入力「sa」は GND、M3S [19:0] の上の GND であるとして示されている。これは 2 つの 0 がこの入力ワードの 2 つの最上位ビットとして追加されることを意味する。そのような追加のビットは適切な 22 ビット幅の入力ワードが適切な符号を有して生成されることを確実にする。

【0179】キャリセーブ加算器 BT2 及び BT3 は 2 つの異なる 22 ビットの入力のキャリワード及びセーブワードを加算して 22 ビット出力セーブワード T3S [21:0] 及び 21 ビットの出力キャリワード T3C [21:1] を生成する。よって、各加算器への入力は

59

88ビット幅であり、各加算器からの出力は43ビット幅である。図10に示したように、キャリアセーブ加算器BT3からの出力と加算される前に、ラッチC10からの出力は最上方のデータ通路におけるラッチC54からの出力と結合される。しかしながら、その「結合(combination)」は上方データ通路におけるそれに続く加算器に到達するまでには必要はない。その結果、図14に示すように、そのシフトされかつ符号拡張された入力値IN0は、上方キャリア出力に接続される。加算器CS0の上方キャリア入力はそのシフトされかつ符号拡張された入力値IN0に接続され、そのシフトされかつ符号拡張された入力値IN1は同一の加算器の上方セーブ入力として接続される。言い換えれば、IN0及びIN1は加算器CS0において後で加算される。

【0180】よって、図10に示した点で必ず行なわれる必要はないが、図10に用いた特定の「ダミー」加算器/減算器240は、動作が行なわれる必要があることを示す。同様に、図10に示した下方のダミー減算器240はラッチC10の出力から減算されるべきラッチC54からの出力を必要とする。これはC10からの出力をC54の出力の補数と加算することと同一である。

【0181】図14を再び参照すると、(図10のラッチC54の出力に対応する)入力IN1の補数は22ビットの入力インバータIN1[21:0] (それはその入力の各ビットのビット単位の論理反転を生成する)によって行なわれる。IN1値の補数、NIN1[21:0]は加算器CS1の上方「セーブ」入力に、シフトされかつ符号拡張されたIN0である対応する上方「キャリア」入力と共に供給される。加算器CS1の上方部分ではIN0引くIN1に相当する減算が行なわれる。

【0182】図10に示した下方の2つのデータ通路においては、分解加算器が共通ブロックBLK 232の出力で上方の2つのデータ通路内に示された分解加算器の代わりに用いられる。各分解加算器又は減算器は分解加算器が後段に続くキャリアセーブ加算器又は減算器と等しい。これは図14に示されている。図10に示した接続構成に従って、減算器CS2及びCS3はそれらの入力としてIN0-IN3の処理した値を有する。

【0183】加算器/減算器C20-CS3の各々からの22ビットのキャリア及びセーブ出力は、分解加算器RES0-RES3のキャリアによって分割される。この分野の当業者には分かるように、キャリア及びセーブ出力の分解はディジタル設計の分野ではよく理解されるので、ここでのより詳細な説明はない。図14に示すように、キャリアセーブ加算器/減算器CS0-CS3のセーブ出力は対応する分解加算器RES0-RES3の「a」入力への22ビットの入力として直接供給される。

【0184】この分野では良く知られているように、2進数の2の補数はそのビットの各々を反転し(「1」を

60

「0」に、又は「0」を「1」に)、 「1」を加算することにより生成される。その「1」についてはビット反転の後に直ちに、又はより遅れて加算することができる。キャリアワードのLSBは常に「0」であり、それはキャリアワード00C及び01Cを分解加算器RES0及びRES1各々に入力されるようにグランドGNDに結合することによって本発明の実施例では行なわれる。しかしながら、2の補数をとった値を生成するために減算器CS2及びCS3のキャリア出力への「1」の加算は、電圧VDDを供給するためにそれらデータワード02C及び03SのLSBを結び付けることにより実行され、それによってキャリアワードの「0」LSBは、「1」による加算と同等のように「1」に置き代わる。

【0185】上記の理由により、「0」は、(LSBをグランドGNDに結合することにより)キャリアセーブ加算器CS0及びCS1からの21ビットのキャリアワードにLSBとして付加される。キャリアセーブ減算器CS2及びCS3からのキャリアワードのLSBは対応するデータラインを供給電圧VDDに結合することにより「1」に等しく設定される。よって、分解加算器RES0-RES3は加算器/減算器CS0-CS3からの出力を分解して22ビットの出力信号OUT0[21:0]~OUT3[21:0]を生成する。

【0186】本発明の実施例におけるIDCT回路の2つの利点については図14に見つけることができる。第1に、制御又はタイミング信号は共通ブロックBLK 232では要求されない。むしろ、その共通ブロックへの入力信号はその共通ブロック232内の純粋な論理演算素子に直ちに供給されて常に処理される。第2に、データワードの適切な調整によって、整数演算(又は少なくとも全ての値のために10進小数点を固定する)を常に用いることができる。これは正確さについて受け入れ難い犠牲を伴うことなく浮動小数点の素子の複雑さ及び遅延を回避する。

【0187】本発明の実施例の他の利点は、示したような入力を順番付けることにより、また平衡の引き合いを用いることにより本発明によれば、シリコンでの実現において多数の点で同様の設計構造を用いることができることである。例えば、図14に示すように、一定係数乗算器MULCIS、MULC3S3、MULC3S2及びMULNCIS全ては同様の構成を有し、データ通路において同一の点でデータを受け入れ、それにより、4つの全ての乗算器が同時に動作することができる。これは「障害(bottlenecks)」を除去し、半導体実現では二重のバレル構造の利点を全て得ることができる。キャリアセーブ加算器BT2及びBT3はそれに続くキャリアセーブ加算器及び減算器のように同様に同時に動作することができる。この設計の対称性及び多数の素子の効率的な同時利用は本発明の実施例による構成全てにおいて共通である。

59

61

【0188】図15は本発明における事後共通ブロック POSTC 233の好ましい構成を示している。図10に示したように、事後共通ブロック POSTC 233の主要な機能は、係数d1, d3, d5及びd7によって共通ブロックの出力を乗算することによりh0~h3値を生成すること、低位出力を生成するためg(k)及びh(k)値を加算すること、及び高位出力を生成するためh(k)値から対応するg(k)値を減算することである。図10及び図15両方を参照して、Bhラッチが活性化されたとき、すなわち制御回路がEN\_BH 10信号を高レベルにし、かつ出力クロック信号OUTC\_\_CLKを高レベルにしたとき、事後共通ブロック POSTC 233は共通ブロックCLK 232からの対応する出力をラッチBH0L, BH1L, BH3L及びBH2Lにラッチする。制御回路が信号EN\_GH及び入力クロック信号IN\_\_CLKを高レベルにすることによりラッチG0L, G1L, G3L及びG2Lを活性化したときg(k), g0~g3値は対応するラッチG0L, G1L, G3L及びG2Lにラッチされる。

【0189】処理した奇数入力、すなわち値h0~h3 20は、EN\_GH及びIN\_\_CLKが高レベルのとき一定係数乗算器D1MUL, D3MUL, D5MUL及びD7MULを介してラッチH0L, H1L, H3L及びH2Lにラッチされる。それらの乗算器はd1, d3, d5及びd7によって各々乗算される。好ましい実施例においては、それらの一定係数乗算器は設計を簡単にしかつ計算速度を上昇させるためには好ましくはキャリセーブ乗算器である。図15に示すように、その一定係数乗算器からの「キャリ(c)」は、以下に述べるある変化を伴って分解加算器H0A, H1A, H3A及びH2A 30の一方の入力に接続される。一定係数乗算器からの「セーブ(s)」は、同様、以下に述べるある強制的な変化を伴って対応する分解加算器の他方の入力に接続される。

【0190】更に、図15に示すように、H0信号のLSBは好ましくはH0についての対応する「セーブ」出力を0にセットすることにより「1」に強制的にされ、第2のビット(H0S[1]に対応する)は「1」にセットされる。一定係数乗算器D3MULのキャリ及びセーブ出力からのデータワードは、同様に分解加算器H1 40Aへの入力として処理される。それら処理及び分解加算器H1Aへの入力の利点。

【0191】本発明によれば、係数乗算器D7MUL及びD5MULからのキャリ出力の全ての22ビットは対応する分解加算器H3A及びH2Aの「a」入力に直接接続される。各乗算器の「セーブ」出力のMSBは対応するデータラインをグランドGNDに結合することにより強制的に「0」にされる。述べたIDCTシステムは上記のCCITT仕様において試験される。デジタル加算器及び乗算器の調整及びその他の公知のプロパテ

62

イのため正確性は10000サンプルを通常無駄にするが、「0」又は「1」へ上記の様々なビットを強制的にすることを行ってデジタル変換の期待誤差を減少させた。データワードのビット処理の結果として、本発明の実施例は同等の正確性を生成するためには24ビットが通常必要であるが、22ビット幅データワードだけを用いてCCITT規格の元で許容できる正確性を達成した。

【0192】正確性の限界や切り捨て及び四捨五入誤差のため、IDCTシステムでは各データワード毎にいくらかの不正確が通常ある。しかしながら、データワードの選択ビットを強制してエラーが全体の結果、統計的によくなるハードウェア内の特定の点で特定のデータワードの中に系統的に生じたことが発見された。ビット強制は例えば、1以上のキャリビットを選択的に所定値に強制することにより処理内で適用しても良い。

【0193】本発明においては、ビット強制方法は特定の値をとるために常に強制されるあるビットと静的である必要はなく、むしろ動的方法が用いられる。例えば、データワードの選択ビットは、ワードが偶数又は奇数、正又は負、閾値より上又は下等のいずれかであるかに従って「1」又は「0」に強制される。通常、単なる小さな系統的な変化は全ての統計的な動作を改善する必要がある。従って、本発明の実施例によれば、選択データワード(必要はないが、好ましくは一度に1ビット及び1ワード)のLSBは「1」又は「0」に強制される。CCITTテストが実行され、その実行のためのCCITT統計値はまとも得られる。そして、そのビットは「1」又は「0」の他値に強制され、テストは再実行される。他のデータワードのLSB(又はLSBs)は「1」又は「0」に強制され、同様の統計値が得られる。様々な強制したワード内の強制したビットの様々な組み合わせに対する統計値を試験することにより、最良の統計的な動作を決定することができる。

【0194】しかしながら、この統計的な根拠を置く改善が必要ないならば、一定係数乗算器D1MUL, D3MUL, D5MUL及びD7MULからの出力が分解加算器H0A~H3Aにおいて従来の方法で分解されても良い。その対応するラッチH0L~H3Lの入力の低位21ビットはそれらの入力のLSBと共にグランドに結合される。

【0195】Hラッチ(H0L~H3L)及びGラッチ(G0L~G3L)からの出力は分解加算器—乗算器S70A, S61A, S43A及びS52Aへのa及びb入力に対生成する。上記したように、ADD信号が高レベルのときそれら入力の加算が行われ、減算可能信号SUBが高レベルのとき「b」入力を「a」入力から減算することが行われる。上方の2つのラッチ対H0L及びG0L, H1L及びG1Lの第2ビットは以下に示す方法において構成を選択することにより処理される。

【0196】分解加算器乗算器S70A, S61A, S43A及びS52Aからの出力は結果ラッチR70L, R61L, R43L及びR2Lにラッチされる。図15bに示すように、加算器/減算器S70A及びS61Aへの入力ワードは、本発明によれば、処理された各入力ワードの第2ビットを有する。例えば、加算器/減算器S70Aの「a」入力への入力ワードの第2ビットはGO[1M], GO[1M], GO[1M], GO[0]である。言い換えれば、第2のビットは値GO1Mを有するためにセットされる。加算器/減算器S70A及びS61Aへの他の入力ワードのビットは同様に処理される。このビット処理は4つの2:1マルチプレクサH01MUX, G01MUX, H11MUX及びG11MUX(図15bの右に図示)によって達成される。本発明においては、それらのマルチプレクサはADD及びSUB信号によって制御される。すなわち、各加算器/減算器S70A及びS61Aがセットされたならば(ADDが高レベル)、第2のビット(H01M, G01M, H11M及びG11M)が1にセットされ、SUB信号が高レベルにセットされたならば、その第2のビットはその実際のラッチ出力値にセットされる。この方法における個々のビットの設定は高速動作を容易に行う。上記したように、数多くのテスト要素ワードの統計的分析が更に正確な結果をそれによって得ることを示している、その好ましい実施例はビット強制構成を含む。しかしながら、この方法においてより小さなワード幅の利点を与えるが、第2のビットを処理する必要はない。

【0197】4つの高位又は低位結果は出力ラッチR70L, R61L, R43L及びR2Lにラッチされる。その結果は多重信号MUX\_\_OUT70, MUX\_\_OUT61, MUX\_\_OUT43, MUX\_\_OUT52の制御の元で最終の出力ラッチOUTFに順次ラッチされる。したがって、結果信号が出力される順はラッチにラッチされる順を変化させることにより簡単に制御することができる。

【0198】事後共通ブロックPOSTC 233内のクロック信号と制御信号との間の関係は図13b及び図13cに示されている。上記したように、2つの1次元IDCT動作は、2次元IDCTを行なうためにデータ間の置き換えで直列的に行なわれる。よって、事後共通ブロックPOSTC 233からの出力信号は、本発明の実施例によれば、まず、RAMメモリ回路等の従来の記憶ユニットに公知の方法で列方向に(又は行方向に)記憶され、そして、次の事前共通ブロックへの入力として供給されるように行方向に(列方向に)記憶ユニットから読み出される。このブロックで上記したように処理されるために共通ブロックBLK 232及び事後共通ブロックPOSTC 233においてもである。

【0199】行(列)で記憶させ、列(行)で読み出すことは、第2の1次元IDCTの前にデータの置き換え

の必要な動作を行なう。第2のPOSTC 233からの出力は所望の2次元IDCT結果であり、それを様々な処理ブロック内で実行された調整シフトをオフセットするためにシフトすることによる従来の方法において調整することができる。特に、1つの位置だけの右シフトは1次元IDCT動作において行なわれる2つの2の平方根の乗算をオフセットするために必要な2による除算を行なう。

【0200】アプリケーションに従ってこの第2のIDCT構造(図11に示されたものと好ましくは同一である)は好ましくは分離の半導体である。これは、回路の単一の実現がリアルタイムで2つの経路を操作することができるよう画素クロックレートが十分であるならば分離した1次元変換が必要ではないが、同一の回路が両方の変換のために用いられるならば起きる速度の減少を避ける。

【0201】図16～図38に示したように、本発明による第2の好ましい実施例は単一の1次元変換を用いる。この実施例は上記したような画素クロックレートの低下を必要としない。第1の好ましい実施例において存在する「分解加算器」は「高速分解加算器」と交換されている。図38に示されたようにそれらは「高速分解加算器」と名称されている。この変換は各データ通路演算ブロックに対してそのデータ入力に更新する時間を与える効果を有する。第1の好ましい実施例において存在する「ラッチ」は2位相の「フリップフロップ」及び「レジスタ」に交換されている。

【0202】その存在する1次元IDCTデータ通路パイプラインの前後に位置されたラッチングメモリ素子は、特に図18に示したように1つのブロックの中に組み入れられている。よって、第2の好ましい実施例の入出力にあるメモリ素子の量はT2データの可変量をバッファとするように増加される。図16及び図17に示されたように、2つのデータストリーム、すなわちストリーム「T1」(未処理のデータ)及びストリーム「T2」(一度1次元IDCTを介し、またTRAM内で置き換えられているデータ)は、時間多量化方式においてデータ通路パイプラインの中に導入される。

【0203】本発明において、各ストリームはデータ通路パイプラインの中にデータ項目のグループを尋ぐためにその変化を用いる。そのデータストリームは、図17、図18及び図3に示すように、それらがデータ通路のパイプライン下方に順次送られるそのデータ通路の出力でデインターリーブされるようにインターリーブされる。グループは数において変化することができ、例においてはそれらは8ビットである。

【0204】本発明においては、T1は遅らせる必要はない。T2がT1とインターリーブ(交互配置)の点に到達したとしても、入力バッファはパイプラインの中にそのデータを導入すべきでない。なぜなら、これはT1の

65

ストリームとぶつかり合い、そして、ストリームT2はデータストリームを遅らせないように余分のバッファリング（蓄え）を備えているからである。しかしながら、その代わりにストリームT1と安全にインターリーブし得るような時間までにその入力ストリームからのデータがバッファに保持される。これは図19及び図33に示されており、ここではストリームT1からのデータは信号「L a t c h 1 (0) ~ L a t c h 1 (7)」を用いてラッチ0~7において第1の変換の中にロードされている。よって、T2からのデータは図33に示した信号を用いて図19に示したように「L a t c h 2 (0) ~ L a t c h 2 (15)」にロードされている。

【0205】インターリーブは「T1 OK2挿入 (i n s e r t)」及び「T2 OK2挿入」信号によって制御される。通常の動作の元ではインターリーブはその信号が高レベルになったとき生じる。しかしながら、「T2 OK挿入」が高レベルの時にT2用のラッチ内のデータ量がまだ適切な量に達していないならば、そのラッチはその機会を失い、データを挿入する次の機会が生じるまでデータのバッファリングを続けなければならない。

【0206】概要として本発明における上記のバッファリングが生じるならば、比較できる「損失 (s l i p p a g e)」がT2の出力で生じなければならない。T2はそのデータ挿入点を失いかつ図19に示したラッチ内にバッファリングし続ける必要があるときスリップする。T2がスリップし、パイプラインの中にデータを導けないならば、データ通路の出力でそのT2ストリーム出力内に対応するギャップが存在する。このギャップはそのT2出力にて余分のバッファリングの使用によって取り去られる。この処理は可変T2-1次元IDCTと共に「固定した」T1-1次元IDCT変換を有すると思われる。ここで、データストリームは同一の演算データ通路パイプライン部分を用いるので時間多重化方式でインターリーブされる。

【0207】本発明においては、データがT1に入力されないとき「復旧 (R e c o v e r y)」が起きる。それはT2バッファがT1及びそのデータストリームに追いつくための機会である。データ無しはIDCTを迂回するデータタイプであり、図34の「L a t c h 2

【φ】」にデータスパイクとして示されている。これは結局はT2入力であるが、それはT2のバッファリングにその出力で満たすことである。「T2 d o u t」信号及び「o u t」信号がサイクルの数だけギャップされるとき復旧は図33及び図25に示される。そのギャップはデータストリームを修復するための基準として用いられる。T2用のラッチがそのデータを挿入することを得ていたときそれら2つの信号の間のサイクルにおけるギャップがバッファリングのギャップと同一である。

66

【0208】POSTC 2333内のTRANSFORM (変換) に続いて、インターリーブされたストリームは図18及び図23に示すように、「T2 o u t」の中でデインターリーブされる。その「T2 o u t」データストリームは上記したようにそのデータ内にスリップギャップを有している。図17に示したT2 o u t [143: φ]は、図17のブロック「IDDPMUX」として示された16:1マルチプレクサブロックに入力する。マルチプレクサブロックは図25に示したように、出力バッファブロック内の16位置のうちの1位置からデータを選択する。この位置は図29に示した制御ロジックによって選択され、その制御ロジックはその入力にてT2のバッファアップをするまでそのギャップを使用する。このギャップは基準として用いられる。マルチプレクサブロックからの出力ストリームT2 D O U Tは修復されたデータストリームである。

【0209】上記したIDCT構造のために本発明の実施例で実行される範囲テストにおいては、中間値及び最終値はC C I T規格に合致する一方、各点で公知の範囲内に維持される。このため、演算計算においてオーバーフロー又はアンダーフローの恐れなく少量だけ（例えば、所望の値に選択データワードのあるビットを強制することによって）上記したように選択値を「調整する」ことができた。

【0210】本発明による方法及びシステムは多数の方法に変更することができる。例えば、加算又は乗算を分解するために用いる構成は公知の技術を用いて変更することができる。よって、好ましい実施例が分離した分割加算器でキャリセーブ素子を用いている減算器の分割加算器を使用することができる。また、本発明の好ましい実施例はそれらの許容範囲内に全ての値が残ることを確実にするように様々な点で下方調整を用いる。しかしながら、下方調整は、他の予防配置がオーバーフロー又はアンダーフローを防止するためにとられるので必要ない。

【0211】本発明の一実施例においては、様々なデータワードの所定のビットがシステム内で要求されたワード幅を減少させるように処理される。しかしながら、様々な中間値は加論ビット処理なく転送されても良い。更に、データワードだけが本発明の図示した例内ではビット処理されるが、一定係数のビットを処理し、C C I T規格の元ではその結果を評価することができる。その結果の比較がある場合に、特定のビットを所定の値に強制することに利点があることを示したならば、対応する乗算器を実行するために要求されるシロコン領域を更に減少させるために、それらの係数の2進数表現における「0」の数を増加させることができる。

【0212】本発明の上述した態様のまとめにおいては、次のことが述べられる。第1データストリーム源を定義する第1のラッチ及び第2データストリーム源を定義する第2のラッチを有するデータを変換する装置。第

67

1及び第2ラッチは1つの演算ユニットと通信する。その演算ユニットは置き換えRAMにデータを伝達し、その置き換えRAMはデータを置換し、それを第2ラッチに転送する。第2ラッチは調整可能であり、受信及び送信されているデータの可変レートを調整するためにサイズを変更できる。第2ラッチ及び第1ラッチは演算ユニットに順次第1及び第2データストリームを伝達する。しかしながら、その第2ラッチの逐次伝達は第1ラッチからの伝達を妨害しない。この方法において、共通の演算ユニットは第1及び第2データストリームのために用いられる。更に、共通の演算ユニットを使用してデータを交換する処理について次に述べる。先ず、第1ラッチにデータを読み込みし、所定数のサイクルに到達すると、演算ユニットにデータを送信し、制御シフトレジスタの中に第1マーカビットを読み込む。次に、第2ラッチにデータを読み込み、その第2ラッチは調整可能であり、異なるレートで受信及び送信されているデータの可変レートを調整するためにサイズを変更できる。次のステップは、第1制御シフトレジスタが所定の状態に達し、かつ第2ラッチが所定データ量で満たされたとき、第2ラッチ内のデータを演算ユニットに送信することである。次に、第2ラッチが所定データ量で満たされないならば、第2ラッチからのデータの送信を防止し、第1ラッチがデータを受け入れていないとき第2ラッチを繰り返させる。

【0213】時間同期についての本発明の詳細な説明 MPEG-2において、ビデオ及びオーディオデータはMPEG-2システムストリームにおいて搬送される情報を用いて同期をとっている。これについて、本質的に同期を処理する2つの情報タイプ、クロック基準及びタイムスタンプがある。クロック基準はデコーダに何番が時間「今(now)」を表すかを知らせるために用いられる。これは、デコーダが現時刻が何であるかを常に知るように規則的な間隔で増加するカウンタを初期化するために用いられる。

【0214】タイムスタンプはプログラム(通常、ビデオ及びオーディオ)を作成するために用いられるデータの流れ(stream)の中に置かれる。ビデオの場合、タイムスタンプは映像と関連され、(クロック基準によって初期化されたカウンタによって設定された)何

「時」に映像が表示されるべきかをデコーダに伝達する。MPEGにおいて、システムストリームの中へは一連のクロック基準で多重化される。それらのクロック基準は「システム時間」を設定する。2つのクロック基準のタイプ、プログラムクロック基準(PCR)及びシステムクロック基準(SCR)である。本発明において、そのクロック基準各々がデコーダによって同一の方法で使用されるので、PCR及びSCRの間の差異は問題とならない。PCR及びSCRは分解能を27MHz(又は秒当たり $1/27 \times 10^6$ )に拡張す

68

る更なるフィールドを有する90MHzの分解能へのタイミング情報を有する。クロック基準は「システム時間」がランダムアクセス又はチャンネル変更の後、再同期化される順でデータストリームの中に含まれる。よって、タイムスタンプは映像を簡単に復号化できるデコーダの仮定モデルを示すことを理解することが重要である。この分野の当業者には理解できるように、実際のデコーダはこれを行うことができず、映像を表示すべき理論的時間を変更するステップをとらなければならない。更に、タイムスタンプ及びクロック基準は表示時に表示時間及び誤差を決定するために用いられる。この変更は特定のデコーダのアーキテクチャの詳細に従う。ビデオデコーダによって導出される遅延はオーディオデコーダにおける遅延と明確に等しくなければならない。

【0215】MPEGを復号するときには「システム時間」の概念において不連続が起きる。例えば、編集したビットストリームにおいて、各編集点は不連続な時間となっている。同様の状態がチャンネル変更でも起きる。タイムスタンプを用いるとき注意しなければならないことは理解できる。なぜなら、その他の体制からクロック基準によって定義された「システム時間」に関して10時間体制において符号化されたタイムスタンプを用いることは不正確な結果を確実に導くからである。

【0216】図39は基本ストリーム250の中にMPEGシステムストリームの多重分離を示している。一般に、ある形式のデータが転送されるものであるが、各基本ストリームはビデオ又はオーディオデータを通常運ぶ。各基本ストリームは一連のアクセスユニットの中に分割される。ビデオの場合、アクセスユニットは映像である。オーディオの場合、オーディオデータの固定サンプル数である。

【0217】また、システムストリームの中へは一連のクロック基準で多重化される。それらのクロック基準は「システム時間」を定義する。本発明によれば、一連のタイムスタンプ251は各基本ストリームと関連されている。そのタイムスタンプは各基本ストリームに対する次のアクセスユニットが与えられる「システム時間」を特定する。それらのタイムスタンプは提示(プレゼンテーション)タイムスタンプ「PTS」として参照される。

【0218】ビデオデータの場合に、タイムスタンプの第2の形式が定義され、デコードタイムスタンプ「DTS」として参照される。PTSが存在しかつPTS及びDTSの間に簡単な関係があるときだけDTSは存在するので、それら2つのタイムスタンプの間の詳細な差異は本発明では影響しないので無視することができる。デコードタイムスタンプ(DTS)はアクセスユニット(ビデオの場合の映像)が復号されるべき時間を定義する。プレゼンテーションタイムスタンプ(PTS)はアクセスユニットが与えられるべき(表示されるべき)時



間を定義する。しかしながら、使用されるタイミングモデルは、デコーダが非常に高速である仮定モデルに用いられる。この場合、PTS及びDTSは互いに等しくなる。

【0219】しかしながら、MPEGビデオの復号においては、複数の映像が記録されている。よって、復号後、映像は表示される前に所定の期間、例えば、数フレーム時間において記憶装置に保持されている。この期間の間、例のその映像に連続して復号される他の映像は表示される。従って、それらの記録された映像については、PTS及びDTSの間には差異がある。

【0220】本発明においては適切に時間に同期するためにタイムスタンプを使用することが必要である。1つの好ましい実施例においては、時間同期回路は映像がそれら復号順に生じるときには復号するパイプライン中の点に位置される。よって、この実施例はDTSを使用する。それでも、その回路は、映像が記録された後に生じるその復号パイプライン内の点に同様に移動させることができ、そのためその映像はそれら表示順に同期回路に到達する。よって、この分野の当業者には理解できるように、PTSがこの実施例では用いられる。

【0221】本発明の好ましい実施例においては、タイムスタンプから得られる情報はトークン(token)によって様々な回路を介して転送される。トークンは一連の1以上の情報のワードからなる。トークンの最初のワードはトークンの種類を識別するコードを含み、よって、情報の種類がそのトークンによって運搬される。現トークン内に多数のワードがあることを示すために「1」にセットされる拡張ビットはトークンの各ワードに関連されている。よって、トークンの最後のワードは「0」である拡張ビットによって示される。本発明においては、トークンの種類を示す最初のワード内のコードは、いくつかのコードが(用いられるべき最初のワード内のビットの残りに他の情報を表させるために)少ビット数を用いる一方、他のコードは多ビット数であるので可変ビット数のものでもよい。

【0222】トークンは制御又はデータトークンのいずれかであるとして特徴づけられる。例えば、システムデコーダとビデオデコーダとの間のインターフェースでは2つの種類の情報、すなわち(1)符号化したビデオデータ及び(2)タイムスタンプ情報から得られる同期時間がある。符号化ビデオデータはデータとして見られ、DATAトークン(例えば、DATA(データ)と呼ばれるトークン)において運搬される一方、同期時間は制御情報として見られ、制御トークン(SYNC TIMEと呼ばれる)において運搬される。また、追加の制御トークンは本発明においては時間から時間へ用いられる。例えば、リセット信号と同様にふるまうFLUSHトークンはエラーのために復号を再度開始する前にビデオ復号回路を初期化するために必要とされる。

【0223】本発明においては、2つの回路の時間同期をとること、特に、2つの回路の第1の回路から第2の回路へシステム時間を直接伝達することなく時間同期をとることが1つの好ましい実施例の目的である。本発明においては、2つの回路の時間同期は、同期した時間カウンタを各回路内に備えることにより第2の回路にシステム時間を直接与えることなく達成される。

【0224】本発明は、各回路に基本ストリーム時間カウンタを備えることにより第1の回路から第2の回路へシステム時間を伝達することなく、システムに2つの回路の時間同期をとることができるようにする。本発明の他の目的は、2つの回路の時間同期をとり、第2の回路へ供給される同期時間を生成するため第1の回路からのタイムスタンプ情報、システム時間及び基本ストリーム時間を用いることにより与えられ、第1の回路内の基本ストリーム時間と同期をとっている第2の回路内の基本ストリーム時間のコピーと比較されているものにおいて、もしあるならばプレゼンテーションタイムエラーを決定することである。本発明のシステムは、システムデコーダからビデオデコーダにシステム時間を直接伝達することなく、各回路内に同期した時間カウンタを備えることによりビデオデコーダにシステム時間を直接供給することなく、また各回路内にビデオカウンタを備えることによりシステムデコーダからビデオデコーダにシステム時間を伝達することなく、システムデコーダ及びビデオデコーダの時間同期をとることができる。

【0225】本発明はシステムにシステムデコーダ及びビデオデコーダの時間同期をとり、ビデオデコーダへ供給される同期時間を生成するためシステムデコーダからのタイムスタンプ情報、システム時間及びビデオ復号時間を用いることにより表示され、システムデコーダ内のビデオ復号時間と同期をとっているビデオデコーダ内のビデオ復号時間のコピーと比較されている映像において、もしあるならば表示タイムエラーを決定することができるようにする。

【0226】本発明においては、タイムスタンプから得られる情報は上記したように、制御トークンを用いてシステムを介して転送することができ、図40は基本ストリームタイムスタンプ管理を行なう本発明の第1の好ましい実施例を示している。システム時間を示すクロック基準253はシステムデマルチプレクサ254によって符号化され、システムデコーダ256内のタイムカウンタ255に必要されるように初期的に供給され、90kHzで増加される。クロック基準の第2のコピーは、基本ストリームデコーダ257の内部にあり90kHzで増加されかつシステムデコーダ256内のタイムカウンタ257と同期をとらているタイムカウンタ258に同時に読み込まれる。

【0227】本発明においては、タイムスタンプ251は入力データに同一量だけ遅延されるようにシステムデ

71

マルチプレクサ254から基本ストリームバッファ260を介して流れる。タイムスタンプ251は基本ストリームデコーダ257の零無し復号時間のために補償するように訂正値を加算される。訂正されたタイムスタンプ251は、基本ストリームデコーダ257の内側のタイムカウンタ258に保持された時間のコピーと比較され、復号化された情報が早過ぎ又は遅過ぎのいずれで与えられるかを判別する。

【0228】上記の実施例は、システムデコーダ内のカウンタが秒当たり90000回変化するので、システムデコーダ256内においてタイムカウンタ255から基本ストリームデコーダ257へシステム時間を直接単に供給するよりも良い。よって、システム時間は全て本質的に基本ストリームデコーダ257に連続的に供給される必要がある。システム時間を連続的に供給することは専用のピン等必要とする。システムデコーダ256内に備えられたタイムカウンタ255及び基本ストリームデコーダ257内に備えられたタイムカウンタ256を用いることにより、システム時間を秒当たり数回だけクロック基準の形式で供給することができる。他の実施例が図41に示されている。図41に示した実施例は基本ストリームデコーダ257にクロック基準253が供給される必要性を避けている。これはシステムデコーダ256及び基本ストリームデコーダ257の両方において維持されている基本ストリーム時間の情報を有している第2のカウンタ「e\_s\_t i m e」262を用いることにより達成される。2つのe\_s\_t i m eカウンタ262及び263は電源投入時に及びチャネル変更等の他の時間にリセットされ、そこから自由に動作し続ける。この実施例は揃った状態である2つのe\_s\_t i m eカウンタ262及び263に頼っている。e\_s\_t i m eカウンタ262及び263が調子を乱さないことを確実にするため能力をみる必要がある。揃った状態に2つのe\_s\_t i m eカウンタ262及び263を維持することを確実にする1つの方法は、図41に示したように、基本ストリームデコーダ257内のe\_s\_t i m eカウンタをリセットするようにシステムデコーダ内のe\_s\_t i m eカウンタからのキャリを用いることである。

【0229】更に図41に示したように、システム時間を示すクロック基準253はシステムデマルチプレクサ254によって復号され、システムデコーダ256内のタイムカウンタ255に供給されて90kHzで増加される。本発明のシステムデコーダ256内のe\_s\_t i m eカウンタ262及び本発明の基本ストリームデコーダ257内のe\_s\_t i m eカウンタ263は互いに同期され、90kHzで増加される。また、基本ストリームタイムスタンプはシステムデマルチプレクサ254によって復号される。よって、同期値Xは基本ストリームタイムスタンプ、タイムカウンタ内に含まれるシステム

72

時間及びシステムデコーダ256のe\_s\_t i m eカウンタ262内に含まれる基本ストリーム時間を用いて式3-1に応じて計算される。

【0230】次の式3-1(a~d)はクロック基準253を基本ストリームデコーダ257に供給することを避ける時間同期のための本発明に応じた1つの方法を示している。式3-1(a)は時間同期のために要求される式である。図41に示したように、基本ストリームデコーダ回路にシステム時間を直接供給することが望まれないので、同期時間表現Xは式3-1(b~d)を用いてシステムデコーダ256によって発生され、この値は基本ストリームデコーダに供給される。同期時間Xは基本ストリームデコーダ257内に設けられたe\_s\_t i m eカウンタ263に含まれる基本ストリーム時間と比較される。従って、その比較結果は復号化情報が早過ぎ又は遅過ぎのいずれで与えられるかを判別するために用いられ、そして更にシステムの時間同期に用いられる。式3-1:

- a) 時間同期 = (基本ストリームタイムスタンプ - システム時間)
- b) 時間同期 = (X - 基本ストリーム時間)
- c) (X - 基本ストリーム時間) = (基本ストリームタイムスタンプ - システム時間)
- d) X = (基本ストリームタイムスタンプ - システム時間 + 基本ストリーム時間)

本発明においては、同期時間Xは基本ストリームデコーダ257の零無し復号時間のために補償すべく訂正値と加算される。訂正された同期時間は、復号化された情報が早過ぎ又は遅過ぎのいずれで与えられるかを判別するために、基本ストリームデコーダ257の内側に設けられたe\_s\_t i m eカウンタ263に含まれる基本ストリーム時間と比較され、そして更にシステムの時間同期に用いられる。時間訂正は、同一の結果のために同期時間Xに加算される代わりに、基本ストリームデコーダ257の内側に設けられたe\_s\_t i m eカウンタ263に含まれる基本ストリーム時間から減算される。上記した実施例は同期時間Xを発生し、情報が早い又は遅いのいずれで与えられるかを判別する解決例である。上記のことを達成する他の多くの等価解決方法があることは当業者には明らかである。

【0231】例えば、図42は本発明に従って、同期時間Xを決定する代替方法を示している。この構成においては、システムデコーダ256は基本ストリーム時間を維持しない。その代わりに、基本ストリームデコーダ257に設けられた基本ストリーム時間カウンタe\_s\_t i m e 263がゼロにリセットされる。直ちにシステム時間をレジスタinitial\_t i m e 256に記録する。e\_s\_t i m e 263内の値は、現システム時間とinitial\_t i m eに記録された値との差異である。よって、システムデコーダ256によって計算すること

ができる。

【0232】次の式3-2(a~c)は時間同期のためのこの代替方法を示している。式3-2(a)は基本ストリームデコーダ257に設けられた  $es\_time263$  に保持された基本ストリーム時間の値を表す式である。これは、システム時間と  $initial\_time$  レジスタ256に記憶された値との関数として同期時間  $X$  を求める式3-2(c)を得るために簡単にされている式3-2(b)を与えるように式3-1(d)に代用されている。

式3-2:

- a) 基本ストリーム時間 = システム時間 -  $initial\_time$
- b)  $X = (\text{基本ストリームタイムスタンプ} - \text{システム時間} + [\text{システム時間} - \text{initial\_time}])$
- c)  $X = (\text{基本ストリームタイムスタンプ} - \text{initial\_time})$

本発明に従った同期時間  $X$  を得るための2つの解決方法は示されている。しかしながら、他の多数の同様の解決方法があることはこの分野の当業者には明らかである。

【0233】図43はビデオタイムスタンプ管理を行なう本発明の他の実施例を示す。システム時間を示すクロック基準253はシステムデマルチプレクサ254によって符号化され、システムデコーダ256内のタイムカウンタ255に必要されるように初期的に供給され、90kHzで増加される。クロック基準の第2のコピーは、ビデオデコーダ270の内側にあり90kHzで増加されるタイムカウンタ258に同時に読み込まれ、システムデコーダ256内のタイムカウンタ255と同期をとられる。

【0234】ビデオタイムスタンプは、入力ビデオデータと同一量だけ遅延されるようにシステムデマルチプレクサ254からビデオ復号バッファ271を介して流れる。ビデオタイムスタンプはビデオデコーダ270の零無し復号時間のために補償するように訂正値を加算される。その訂正されたビデオタイムスタンプはビデオデコーダ270の内側のタイムカウンタ258の時間のコピーと比較され、復号化された映像が早過ぎ又は遅過ぎのいずれで与えられるかを判別する。

【0235】図43に示した実施例は、システムデコーダ内のカウンタが秒当たり90000回変化するので、システムデコーダ内においてタイムカウンタからビデオデコーダへシステム時間を直接単に供給する処理を改善する。よって、システム時間は全て本質的にビデオデコーダに連続的に供給される必要がある。システム時間を連続的に供給することは専用のピン等を必要とする。システムデコーダ内に備えられたタイムカウンタ及びビデオデコーダ内に備えられたタイムカウンタを用いることにより、システム時間を秒当たり回数だけクロック基準の形式で供給することができる。

【0236】図44を参照すると、システム時間を示すクロック基準はシステムデマルチプレクサ254によって符号化され、システムデコーダ256内のタイムカウンタ255に供給され、90kHzで増加される。システムデコーダ256内の  $vid\_time$  カウンタ272及びビデオデコーダ270内の  $vid\_time$  カウンタ273は互いに同期をとられ、90kHzで増加される。ビデオタイムスタンプは、式3-3に応じて、システムデマルチプレクサ254によって復号化される。よって、同期値  $X$  はビデオタイムスタンプと、タイムカウンタ273内に含まれるシステム時間と、システムカウンタ256内の  $vid\_time$  カウンタ272に含まれるビデオ復号時間とを用いて計算される。

【0237】次の式3-3(a~d)は、クロック基準をビデオデコーダ270に供給することを避ける時間同期のために本発明に応じた1つの方法を示している。式3-3(a)はタイム同期のために要求される式である。図44に示したように、ビデオデコーダ回路270にシステム時間を直接供給することが望まれないので、同期時間表現  $X$  は式3-3(b~d)を用いてシステムデコーダ256によって発生され、ビデオデコーダ270に供給される。同期時間  $X$  はビデオデコーダ270内に設けられた  $vid\_time$  カウンタ273に含まれるビデオ復号時間と比較される。その比較結果は復号化映像が早過ぎ又は遅過ぎのいずれで表示されるかを判別するために用いられ、そして更にシステムの時間同期に用いられる。

式3-3:

- a) 時間同期 = (ビデオタイムスタンプ - システム時間)
- b) 時間同期 = ( $X$  - ビデオ復号時間)
- c) ( $X$  - ビデオ復号時間) = (ビデオタイムスタンプ - システム時間)
- d)  $X = (\text{ビデオタイムスタンプ} - \text{システム時間} + \text{ビデオ復号時間})$

本発明においては、同期時間  $X$  はビデオデコーダの零無し復号時間のために補償すべく訂正値と加算される。訂正された同期時間は、復号化された映像が早過ぎ又は遅過ぎのいずれで表示されるかを判別するために、ビデオデコーダ270の内側に設けられた  $vid\_time$  カウンタ273に含まれるビデオ復号時間と比較され、そして更にシステムの時間同期に用いられる。時間訂正は、同一の結果のために同期時間  $X$  に加算される代わりに、ビデオデコーダ270の内側に設けられた  $vid\_time$  カウンタ273に含まれるビデオ復号時間から減算され得る。上記した本発明の実施例は同期時間  $X$  を発生し、映像が早い又は遅いのいずれで表示されるかを判別する解決例である。しかしながら、上記のことを達成する他の多くの等価解決方法があることはこの分野の当業者には明らかである。

【0238】本発明における他の特徴は完全な33ピッ

トのタイムスタンプ数又は42ビットのクロック基準数を処理する必要がないことである。本発明はビデオデコーダ270で16ビットを扱うことを可能にするためカウンタを16ビットに制限する。一見では16ビットは90kHz(使用されるべき2/3秒だけ)の分解能で十分な数の範囲を表わすことができないように見える。しかしながら、ビデオデコーダ270における時間制御はフィールド時間にだけ正確であるのであるのでそのような高い精度は要求されない(ビデオタイミング発生器V10 TGは自由に動作するか、又は復号化されるべきMPEGストリームと共に進めることは何もないものに同期結合されるので)。よって、それはタイムスタンプ又はプレゼンテーション(表示)時間とは関係しない。

【0239】図44及び図45に示したように、同期時間Xはビデオデコーダ270内のVid\_timeカウンタ273は16ビットだけを用いている。これは2つのフックタによって可能にされる。まず、システム時間とタイムスタンプ(同期時間を得るために用いられる：式3-3を参照)との間の差異は常に小さくなるべきであり、それにより最上位ビットの除去が可能となる。第二に、最も近いフレーム時間にビデオの表示を制御することができるだけである。よって、最下位ビットは必要とされず、4ビットだけ右へシフトすることによって除去される。

【0240】従って、本発明に用いられる時間情報の16ビットは約180μs(フィールド時間の約1%)の精度をもって1.5秒程までのタイミングエラーを処理することができる。PAL又はSECAM方式のヨーロッパの625ラインのテレビジョンシステムは5625

Hzのクロックの112.5ティック(ticks)であり、NTSC方式の525ラインのテレビジョンシステムは93.84ティックである。よって、16ビットを用いることはフィールド時間の約1%の精度でタイミング計算を可能にする。

【0241】図46はハードウェアを介してタイムスタンプを移動する本発明に応じた好ましい処理を示している。このハードウェアにおいて情報を伝達する好ましい方法はトークンであるが、代替方法を用いても良いことは明らかである。ハードウェアは2つのモジュールに分割される。第1のモジュールは、上記したように同期時間Xを含むトークンSYNC\_TIMEを生成する原因になっており、これは関連したPICTURE\_\_STARTトークンの前に起きる。MPEGシステムストリームにおいて、タイムスタンプはパケットヘッダ内に搬送され、データパケット内の第1の映像について示す。そのパケットはビデオデータと共に整っていないので、一般に、タイムスタンプが示している映像の開始の前に、以前の映像の終わりがあ

る。【0242】同期時間情報はマイクロプロセッサインターフェースを介して又はトークンを用いて本発明の実施例において供給されるようにしても良い。いずれの場合も、表12に更に詳細に示したように、同期時間データ(16ビット)は(アクセスを各バイト個々にさせるために2つの部分に分割された)同期時間レジスタに保持される。

【0243】

【表12】

レジスタ	サイズ/bit	リセット	説 明
ts_low	8/rv	—	同期時間値の下位8ビット。 ts_lowレジスタは以前の書き込み値に影響を与えることなく新たな値がこのレジスタに書き込まれるように従属動作する（それはSYNC_TIMEトークンの部分になる）。 ts_lowレジスタへの書き込みはマスタレジスタに影響を与える一方、読み出しはスレーブレジスタに読み戻す。マスタからスレーブへの転送はts_validを用いて効果を上げるまでts_lowに書き込まれる値を読み戻せることはできない。
ts_high	8/rv	—	同期時間値の上位8ビット。 ts_lowのように同一の方法で従属動作する。
ts_valid	1/rv	0	このビットはts_low及びts_highのマスタスレーブ転送を制御する。 値がts_lowts_highに書き込まれたときマイクロプロセッサはこのビットに値1を書き込むべきである。それは値1を読み戻すまでそのビットを得るべきである。この点においてts_low及びts_highに書き込まれる値はスレーブレジスタに転送され（読み戻すことができ）、ts_validは1にセットされる。マイクロプロセッサは次のアクセスのための準備中で値0を書き込むべきである。

ts_waiting	1/ro	0	0のセット時にレジスタts_low及びts_highは有効な同期時間情報を含まない。 1のセット時にレジスタts_low及びts_highは有効な同期時間情報を含む。SYNC_TIMEトークンは次のPICTURE_STARTトークンの前に発生され、ts_waitingは0になる。 このビットは以前の同期時間値がマスタスレーブ転送によって上書きされる前に用いられたことを確実にするためにts_valid内に1を持つ前にそれが0であること確実にするために得られべきである。
------------	------	---	---

表12 同期時間を処理するマイクロプロセッサレジスタ

本発明において、フラグts\_waitingは、有効な同期時間情報がタイムスタンプレジスタにあるという事実を示すためにセットされる。データがSYNC\_TIME

40 トークンを用いて供給されたならば、そのトークンはトークンのストリームから取り去られる。  
【0244】 PICTURE\_STARTトークンが活性化されるとき同期時間レジスタの状態を示すフラグは試験される。そのフラグがセットされていないならば、動作はなく、PICTURE\_STARTトークン及び全てのその後のデータは影響されない。しかしながら、そのフラグがセットされているならば、有効な同期時間情報が利用できることを示し、それでSYNC\_TIME

50 トークンはPICTURE\_STARTトークンの前の

データストリームにおいて生成されかつ含まれる。そして、そのフラグがクリアされ、同期時間レジスタが次に生じるタイムスタンプのために利用できるようになる。

【0245】 図40に示したように第2のモジュールは27kHzのクロックで動作する事前調整器（プリスケール）と、マイクロプログラムابل状態マシン（MSM）218と連動される事前調整器278によるクロックで動作するvid\_timeカウンタとからなる。図44及び図46に示したように、4800によってクロックを分割する事前調整器278が備えられている。言い換えれば、4800は16を300（27MHz/90kHz）倍である。図45及び図46に示された4804、8のオプションは以下に示す。

【0246】NTSC方式のカラーテレビジョンにおいて、フレームレートは30Hzではなく、実際にはほぼ29.94Hz（正確には30000/1001Hz）である。[カラーテレビジョンの出現前には正確に30Hzが用いられた。]NTSC方式のラインタイム（ラインタイムはフレーム時間の1/525である）当たりクロック周期は正確に1716.27MHzである。米国テレビジョン規格団体は将来30Hz（HDTVでは60Hz以上）に戻ることに興味を示している。その結果、MPEGは正確な30Hzのフレームレートをサポートしている。しかしながら、27KHzのクロック（ライン当たり1714.29…サイクルである）から安定した30Hzのテレビジョン信号を発生することはできないので、MPEGのフレーム形成に30Hzのラスタを生成することは困難である。

【0247】1つの可能な解決方法は、27MHzのクロックを発生する代わりに27.027MHzのクロックを発生するようにデコーダでクロックレートを変化（bend）させることである。このクロックは90Hzのクロックを生成するためMPEGのクロック基準を用いて（300よりむしろ）300.3の分割器で発生される。この27.027MHzクロックは、27MHzから29.94Hzのフレームレートを与える同等のビデオタイミング回路のクロック動作が正確な30Hzレートを与えるときである。

【0248】本発明におけるフレーム形成においては、90kHzは16の更なるファクタによって事前調整される。よって、300.3×16=4804.8による27.027MHzのクロックの分割である。vid\_timeカウンタ273（上記した）はビデオ復号時間を含み、事前調整器278が最終計数値に達する毎に増加される。そのvid\_timeカウンタ273はリセットタイムビンによってリセットされる。

【0249】本発明における事前調整器及びvid\_timeカウンタについては、他で用いられているレジスタフィードバック（resistive feedback）又はウィークフィードバック（weak feedback）よりもα粒子墜落（corruption）に更に抗する完全なクロック動作のフィードバックフリップフロップで実現することができる。時間カウンタを用いることは、ビデオ復号デコーダ内の時間カウンタがシステムデコーダ内の時間カウンタと調子を揃えて動作することを確実に助ける。

【0250】図47は、MSM218がSYNC\_TIMEトークンを入力したとき行なう処理を示している。MSM218はビデオ時間カウンタによって示される現在の時間を読み取り、そしてそれとビデオSYNC\_T

IMEトークンによって供給される値とを比較することができる。よって、映像を復号化する状態であるべき時間と比較したように、現在の時間が早いか遅いかを判別することができる。

【0251】本発明においては、16ビット符号のタイムスタンプの訂正値は、ビデオSYNC\_TIMEトークンによって搬送された同期時間Xに加算される。この訂正値はチップリセットにてMSM218によってリセットされ、動作がないならば、同期時間は変更されないで残る。制御マイクロプロセッサは同期時間を修正するためにts\_correctionの中に値を常に書き込み、ビデオ及びオーディオデコーダを介して差異を示す遅延を補償する。

【0252】vid\_timeカウンタ273内に含まれる現ビデオ復号時間は訂正された同期時間から減算される。値の符号はその誤差（エラー）の方向であり、（もしあるならば、MSM218によって発生されるエラーコードを決定する。）その誤差の絶対値がとられ、その結果は、タイミングエラーが許容限界値内にあるか否かを判別するために閾値と比較される。その各目時間からフレーム時間をプラス又はマイナスした精度にビデオタイミングを制御することができるので（VTG33が自由に動作している）、この閾値は1つのフレーム時間にてセットされる。

【0253】その誤差がフレーム時間を越えたならば、訂正値が作成されなければならない。本発明のMSM218は適切な時間まで復号を簡単に遅延させることができるので、復号化が早過ぎるならばMSM218はその状態をそれ自身で訂正することができる。しかしながら、復号化が所望の時間より遅れている場合には符号化データバッファの出力で映像を確実に除去することができないので、時間訂正は難しい。本来はそのシーケンスの復号化は中断されており、その状態を訂正する最も確実な方法は、ランダムアクセス又はチャンネル変更と同様の方法で復号化処理を再度開始することである。この処理を容易にするためには、適切なスタートコード又はFLUSHトークンが活性化されるまで全てのデータを除去するようにMSM218の制御レジスタをプログラムすることである。更に、エラー「ERR\_TOO\_EARLY」はdisable\_too\_earlyの設定とは無関係にスタートアップ（始動）の間発生されない。なぜなら、スタートアップの際、第1の映像は早くあることを期待されるからである。

【0254】表13はMSM218のレジスタがどのように動作するのか示し、レジスタの中に置かれた動作及びエラーメッセージ情報の詳細を示している。

【0255】

【表13】

レジスタ名称	サイズ	リセット状態	説明
ts_correction	1b/rw	ゼロ	それを使用する前に同時間に加算される訂正値。
frase_time	1b/rw	228 or 188	映像を復号化するタイミング上の許容差を表す。PAL/NTSCによって定められた状態にリセットする。
vid_time	1b/ro	ゼロ	reset 又は reset_time のいずれかによってリセットする。ビデオ復号化時間のその現在値。
manual_startup	1/rw	ゼロ	1へのセット時にはスタートアップはdecode_disableを用いて通常に行なわれることである。この場合に、MSMではSEQUENCE_END及びFLUSHトークンはdecode_disableを1にセットさせる。し
decode_disable	1/rw	ゼロ	0へのセット時には復号化が普通に進められる。 各映像の開始時にMSMはdecode_disableの状態をチェックし、1にセットされているならば進めない。 マニュアルスタートアップが行われるならば(すなわち、タイムスタンプ管理ハードウェアがなく)、manual_startupが1にセットされることと同一の時間にこのビットは1にセットされるべきである。
disable_too_early	1/rw	ゼロ	1へのセット時には早過ぎの復号化を示すエラー「ERR_TOO_EARLY」は抑制され、MSMはその状態を修復することを単に待つ。
NTSC_80	1/rw	ゼロ	1へのセット時には事前調整器は4800よりむしろ4804.8により分割する。30Hzのフレームレートの復号化のとき自動的にセットされる。

レジスタ名称	サイズ	リセット状態	説明
discard_if_late	1/rw	ゼロ	これは「ERR_TOO_LATE」が発生されない限り効果を持たない（又はエラーがマスクされなかったならば発生される）。1にセットされるならば、discard_until によって示される状態までデータは除去される。
discard_until	2/rw	ゼロ	符号化を開始させたタイムスタンプを終了させる状態を示す。 0-FLUDH 1-SEQUENCE_START 2-GROUP_START 3-NEXTPICTURE 1つの映像の符号化はその映像が交互のトップ/ボトムフィールド構造を保存するためにダミーフィールドの発生によるフィールド映像ならば、直ちに行われない。その結果としてdiscard_until が「Next Picture」にセットされるが、そのダミーフィールドが発生されるならば1つの更なる映像が除去される。

表13 タイムスタンプMSMレジスタ

本発明の同期時間処理の結果として、2つのエラーのうちの1が発生可能となる。ERR\_TOO\_EARLYは、復号化がタイムスタンプによって示される時間より早くに起きているならば、発生される。ERR\_TOO\_EARLYは抑制されるが、ERR\_TOO\_LATEは全てのエラーがマスクされない限り常に発生される。

【0256】まとめにおいて、本発明は、表示時間を決定するタイムスタンプと、第1の回路内のシステム時間を初期化するクロック基準と、第1の回路内のシステム時間を維持するクロック基準との伝達をなす第1時間カウンタと、その第1時間カウンタと同期をとった第2の回路内のクロック基準によって初期化される第2時間カウンタとを有して時間の同期をとり、システム時間のローカルコピーを維持し、タイムスタンプと第2時間カウンタとを比較することによりシステム時間のローカルコピーとシステム時間との間の表示タイミングエラーを決定する装置を備えている。それは更に、表示時間を決定するためにタイムスタンプと、システムデコーダ内のシステム時間を初期化するクロック基準と、システムデコーダ内のシステム時間を維持するクロック基準との伝達をなす第1時間カウンタと、その第1時間カウンタと同期をとった第2の回路内のクロック基準によって初期化される第2時間カウンタとを用いてシステムデコーダ及びビデオデコーダとの同期をとり、システム時間のロー

カルコピーを維持し、タイムスタンプと第2時間カウンタとを比較することによりシステム時間のローカルコピーとシステム時間との間の表示タイミングエラーを決定する装置を備えている。その他の実施例は、第1の回路内のシステム時間を初期化するクロック基準を用いて第1の回路及び第2の回路の同期をとる装置を備え、第1の回路はシステム時間を維持するためのクロック基準との伝達をなす時間カウンタを有し、基本ストリーム時間を与えるために第1の回路内に第1基本ストリームタイムカウンタを含む。第1の回路は基本ストリーム時間とタイムスタンプとを加算しシステム時間を減算することにより同期時間を発生する。第2の回路は第1の回路からの同期時間を入力するように適合され、基本ストリーム時間のローカルコピーを与え、かつ同期時間と基本ストリーム時間のローカルコピーとを比較することによりシステム時間とタイムスタンプとの間のタイミングエラーを決定する第1基本ストリーム時間カウンタと同期した第2基本ストリーム時間カウンタを有する。このように、クロック基準信号はタイミングエラーを決定するために第2の回路に直接供給される必要はない。他の実施例においては、第1の回路及び第2の回路の同期をとる装置は第1の回路内のシステム時間を初期化するクロック基準を有している。第1の回路はシステム時間を維持するクロック基準との伝達をなす時間カウンタと、ビデオ



オ復号時間を与える第1ビデオ時間カウンタとを有している。その第1の回路はビデオタイムスタンプを入力する用に適合され、ビデオ復号時間をビデオタイムスタンプに加算しシステム時間を減算することにより同期時間を発生する。第2の回路は第1の回路からの同期時間を入力するように適合され、ビデオ復号時間のローカルコピーを与え、かつ同期時間とビデオ復号時間のローカルコピーとを比較することによりシステム時間とビデオタイムスタンプとの間のタイミングエラーを決定する第1ビデオ時間カウンタと同期した第2ビデオ時間カウンタを有する。よって、クロック基準信号はタイミングエラーを決定するために第2の回路に直接供給される必要はない。本発明は、またバケットヘッダ内にタイムスタンプを有するビデオデータストリームを備えることによりタイミング情報を与える方法を含み、そのタイムスタンプはデータのバケットの内の最初の映像に関している。次のステップにおいて、レジスタはバケットヘッダから取り出されレジスタ内に保持される有効なタイムスタンプ情報を示すために用いられるフラグを備える。次に、タイムスタンプはビデオデータストリームから取り去られてレジスタ内に保持される。次いで、その方法は映像開始となり、それに続いてフラグ状態をチェックすることによって有効なタイムスタンプ情報がレジスタに含まれるか否かを判別するためレジスタの状態を試験する。有効なタイムスタンプ情報がレジスタ内に含まれるならばタイムスタンプは映像開始に充当して発生され、そして、そのタイムスタンプはデータストリーム内に戻される。他の実施例は上記したように、基本ストリーム時間カウンタが16ビットに制限される装置を含む。同様に、上記したように、基本ストリームデコーダ内に設けられた第2基本ストリーム時間カウンタが16ビットに制限される装置が備えられている。更に、上記したように、同期時間は基本ストリームデコーダを制御するために16ビットにされる装置が備えられている。また、本発明はビデオを復号し、属値に対する表示時間エラーを決定する処理を有する。それは更なる処理を行い、タイムスタンプトーンが示されているか否かを判別し、そのタイムスタンプトーンをビデオデータと比較するためにトーン内のビデオデータを分析し、タイミングエラー量を判別するために比較した値を発生する。次に、属値に対して比較されたときに比較した値がタイミングエラーが示されたときの許容パラメータ内にあるか否かが判別され、その比較した値が許容パラメータ以外にあるときを示す。代替実施例はシステムデコーダとビデオデコーダとを用いる装置を含む。そのシステムデコーダはMP EGシステムストリーム及び多重分離ビデオデータとそのストリームからのビデオタイムスタンプを受け入れるように適合される。そのシステムデコーダはシステム時間を示す第1時間カウンタを有する。ビデオデコーダはほぼ一定のレートでビデオデータを受け入れ、かつ可

変レートでそのビデオデータ出力し、ビデオタイムスタンプを供給するビデオデコーダバッファを有する。ビデオデータから映像を復号するビデオデコーダは復号された映像のためのビデオタイムスタンプを適切な表示時間を決定するように第2時間カウンタと比較する。第1の回路にシステム時間(SY)、タイムスタンプ(TS)及び基本ストリーム時間(ET)を与えることにより第1の回路と第2の回路との間の時間エラーを決定し、基本ストリーム時間(ET)、タイムスタンプ(TS)及びシステム時間(SY)を用いることによって同期時間(X)を得て、式 $X = ET + TS - SY$ に応じて第2の回路に同期時間(X)を与えかつ同期した基本ストリームタイム(ET2)を発生し、同期した時間を用いることによって同期エラーを得る方法が備えられ、よって、第2の回路にシステム時間を供給することなく、式 $ET2 - X$ に応じて第1の回路は第2の回路に同期した時間にすることができ。第1の回路と第2の回路との間の時間エラーを決定する他の方法は次のステップを有している。すなわち、第1の回路にタイムスタンプ(TS)及び初期時間(IT)を与える、タイムスタンプ(TS)及び初期時間(IT)を用いることによって同期時間(X)を得て、式 $X = TS - I$ に応じて第2の回路に同期時間(X)を与えと共に同期した基本ストリーム時間(ET)を発生し、同期した時間(X)を用いて式 $ET - X$ に応じて時間エラーを得る。このように、第1の回路は、第2の回路にシステム時間を供給することなく、第2の回路に同期した時間にすることができ。更に、第1の回路と第2の回路との間の時間エラーを決定する他の方法は次のステップを有している。すなわち、第1の回路にシステム時間(SY)、ビデオタイムスタンプ(VTS)及びビデオ復号時間(VT)を与え、ビデオ復号時間(VT)、ビデオタイムスタンプ(VTS)及びシステム時間(SY)を用いることによって同期時間(X)を得て、式 $X = VT + VTS - SY$ に応じて第2の回路へ同期時間(X)を与えると共に第1の回路内のビデオ復号時間(VT)に同期した第2の回路内のビデオ復号時間(VT2)を発生し、同期した時間(X)を用いかつ式 $VT2 - X$ に応じて時間エラーを得る。よって、第1の回路は、第2の回路にシステム時間を供給することなく、第2の回路に同期した時間にすることができ。

【0257】非同期スイングバッファリングについての本発明の詳細な説明

非同期スイングバッファリングについて本発明においては、2つのバッファが非同同期で動作される。一方は書き込まれ、他方は読み出される。よって、これは、処理能力の最初のレートに有するデータストリームに対して他のレートに再同期させるが、一方では所望のレートを維持させる。本発明において、書き込み制御及び読み出し制御の両方はそれらが使用しているバッファ転送のため

にその制御がアクセスを持っているのか又はそのバッファを実際にアクセスしているかどうかを示す状態指示器(インジケータ)を有する。各側(サイド)はその他方の側へバッファを使用していることを示すため単一ビットを転送する。これは非同期回路の2つの側面と同期をとるために必要な唯一の信号である。

【0258】1つの制御回路は(読み出し又は書き込み)はバッファへのアクセスを終了したとき本発明は制御を他方の回路に与える。制御がスイングされた後、2つの制御回路が同一のバッファを用いることを試みるならば、後の制御は待機を開始することになる。制御回路は各側が交互にバッファを用いるまで、すなわち他方の側がスイング状態となるまで待機する。スイングの後、他方の側に交互にバッファを使用していることを分かるならば、待機することなく直ちにアクセスを開始する。このバッファ間で件数するシステムは同一のバッファ、この場合バッファ0を用いて両バッファによって始動する。読み出し側は待機によって始動され、一方、書き込み側はいずれのバッファからも読み出すために有効なものはないのでアクセスする。

【0259】本発明の実施例においては、スイングバッファは両側によってアクセスされているバッファに対応した全ての信号、すなわちアドレスストロブ、アドレス、及び読み出し又は書き込み側から多重化されたデータを有する2つのディスクリットRAMである。この構成は2つのバッファ間において多数の信号をバス転送する多数の領域を使用することを示している。1つの構成の中に2つのRAMを結合されることはバス転送する領域を汎用を節約する一方、動作を同一標準にしよう。この構成は本発明の第1の実施例のディスクリットRAMのうちの1つの2倍のセル行を含む。しかしながら、第2の実施例はディスクリットバッファについての読み出し及び書き込みが同期して又は非同期で起きているので2つの対のビット行を持たなければならない。アクセスがそのディスクリットRAM用と同一の幅であるので各行は元々の幅(すなわち、同一のセル数)である。各行の行は同一の幅であるようにアクセスされるが、異なるバッファからそれでは異なる対のビットラインに接続する。同一のアドレスを用いてそれら対の行は読み出しアドレスに接続された1行のデコーダ及び書き込みアドレスに接続された1行のデコーダによって容易にアクセスされ得る。その読み出し及び書き込み制御は行デコーダによってアクセスされた対について衝突が起きないように同一の時間に同一のバッファと決してアクセスしない。

【0260】各行デコーダが各バッファから行をアクセスできる同じ方法で、本発明の構成内の読み出し及び書き込み回路の間には各バッファから各対のビットライン、1対に接続する。その読み出し及び書き込みはバッファ各々に多重化され、上記した同一の理由のためそれ

らは衝突しない。図48に示すように、本発明によれば、スイングユニット1はRAM12及び14を有するスイングバッファ10を含む。また、そのスイングバッファ1はRAM12及び14内の又はからのデータを制御する書込制御回路及び読出し制御回路を含む。読出し制御回路及び書込制御回路はストロブ、データ及びアドレス制御ライン8の使用によりこれを達成している。ライン7及び9は書込制御回路により使用されるRAM及び読出し制御回路により使用されるRAMを示すための制御ラインである。ライン7は書込制御回路を制御するように機能し、例えば、読出し制御回路がRAM12を用いるとき低レベル、RAM14を用いるとき高レベルになる。同様に、ライン9は読出し制御回路を制御するように機能し、例えば、書込制御回路がRAM12を用いるとき低レベル、RAM14を用いるとき高レベルになる。

【0261】本発明において、スイングバッファは2つのRAMアレイ12及び14を有する。スイングバッファ10は、装置をメモリの高速アクセスに必要なバンド幅にすることができるようになるRAM領域に対して非同期で交互に読み出し及び書込を行うことができる。RAM12及び14は14次の信号を必要とする。すなわち、書き込みアドレス16、読み出しアドレス18、データ入力20、データ出力22、読み出し及び書き込み可能信号(図示せず)である。図49も参照のこと。

【0262】書き込みアドレス及び読み出しアドレス信号はマルチプレクサ24によって多重化される。RAM12及び14は従来のセンスにおいて書込回路、行デコーダ及び読出し回路と共に動作する。本発明の第1の実施例においてスイングバッファ10の初期化期間には、RAM12は、制御回路が書込可能信号をRAM14に切り換えるまで書き込み状態となる。

【0263】RAMアレイ12は書き込まれると、それは読出し回路4の制御の元で読み出され始めることになる。この時間の間、RAMアレイ14は書き込まれる。RAMアレイは読出しアレイ制御又は書込アレイ制御の元で動作を始めるときその制御は読み出し又は書き込みが完了するまで行われて、そして制御は逆に変わる。読み出し制御回路4が12等のRAMアレイをアクセスし、書き込み制御回路2が同一のRAMアレイ12をアクセスする必要がある状態において、書き込み制御回路は待機を開始する。

【0264】よって、本発明においては、2つの制御イベントが作成される。書込制御回路又は読出し制御回路が異なるRAMにスイング対応しているときRAMが自由であり、選択する回路の制御下でないでそれはRAMを直ちにアクセスし始めるか、又は待機し始める。始動の間にはいずれのバッファからも読み出すために有効なものはないので読み出し側は書き込み側を参照する。

【0265】本発明の第2の実施例は図50に示されて

いる。統合シングバッファ 50 は RAM アレイ 14 と結合された RAM アレイ 12 の論理サイズを有する RAM アレイ 32 を含む。言い換えすれば、第 1 及び第 2 の実施例両方において同一の RAM の量があるが、第 2 の実施例においては結合されている。よって、統合シングバッファは多数のバス領域を節約するという利点を有する一方、同一のシングバッファ機能を行なう。

【0266】本発明の第 2 の実施例において、書込回路及び読出回路 34 及び 36 はシングバッファ 10 内に用いられるものと同様である。しかしながら、それらの回路は以下に述べる対のビットラインから選択するセレクタを含んでいる。同様に、読出アクセス行デコーダ 38 及び書込アクセス行デコーダ 40 はシングバッファ 10 内に含まれるものと同様である。しかしながら、それらは図 51 に述べられているように対の行をアクセスすることができる。

【0267】図 51 に示すように、本発明における統合シングバッファ 30 の特定の構成は詳細に説明されている。個々のセル 42 は行 44 に含まれる。読出行デコーダ 38 及び書込行デコーダ 40 は対にて行 44 をアクセスする。1 対の行はアドレスライン 16 及び 18 によって与えられる同一のアドレスを有する。読出バッファライン 52 及び書込バッファライン 54 は対の行 42 のうちの 1 つを選択するための制御情報を与える。バッファ 0 ビットライン 48 及びバッファ 1 ビットライン 50 はセルの交互の行に接続され、また読出及び書込回路 34 及び 36 に接続されている。アドレス指定の図示において明確化のために、明斜線はバッファ 0 の行をアクセスする読出行デコーダ 38 を示し、同様に、暗斜線はバッファ 1 の行をアクセスする書込行デコーダ 40 を示している。

【0268】まとめとして、本発明は、少なくとも 2 つの RAM アレイと、RAM アレイ内へのデータ入力を制御するため RAM アレイとの伝達をとる書込制御回路と、RAM アレイ内からのデータ出力を制御するため RAM アレイとの伝達をとる読出制御回路とを有するシングバッファ装置を備えている。更に、書込制御回路及び読出制御回路は RAM アレイの同期した制御を可能にするために互いに伝達をとる。また、シングバッファ装置は RAM アレイと、1 対のビットラインを介して RAM アレイとの伝達をとる書込制御回路と、他の 1 対のビットラインを介して RAM アレイとの伝達をとる読出制御回路と、個々のセルが読み出されるように 1 対の行を介して RAM をアクセスする読出行デコーダ及び書込行デコーダとを有する。また、本発明は少なくとも 1 対の行を復号するデコーダを用い、アクセスされるべき行のうちの 1 を選択し、読出回路及び書込回路に接続された少なくとも 2 対のビットラインを用い、用いられる

その対のビットラインを選択して、RAM 内の少なくとも 1 対のセルを復号化することにより、RAM 非同期でアクセスする方法を提供する。

【0269】ビデオ情報の記憶についての本発明の詳細な説明

ビデオ伸張 (decompression) システムは映像情報を復号しかつ表示する 3 つの基本部分を含む。そのビデオ伸張システムの 3 つの主要部分は空間 (spatial) デコーダと、一時 (temporal) デコーダと、ビデオフォーマッタとである。本発明は一時デコーダと、ビデオフォーマッタと、その一時デコーダ及びビデオフォーマッタがそれぞれの映像バッファ (以下フレーム記憶バッファ) を管理する方法とを含んでいる。MPEG システムにおいて、一時デコーダは 2 つのフレーム記憶バッファを含み、ビデオフォーマッタは 2 つのフレーム記憶バッファを含んでいる。

【0270】MPEG は 3 つの異なる種類の映像を使用する。すなわち、イントラ: Intra (I)、予測: Predicted (P)、双方向補間: Bidirectionally interpolated (B) である。B 映像は他の 2 つの映像からの予測に基づいており、1 の映像は未来からであり、1 つは過去からである。I 映像については一時デコーダによって更なる複合化の必要はないが、P 及び B 映像の復号化において後の使用のため 2 つのフレーム記憶バッファのうちの 1 つに記憶されなければならない。P 映像の復号化は以前に復号化した P 又は I 映像から予測を行なう必要がある。その復号化された P 映像は P 及び B 映像を複合化する際に使用するためにフレーム記憶バッファに記憶される。B 映像は両方のフレーム記憶バッファから予測することができる。しかしながら、B 映像はフレーム記憶バッファに記憶されない。

【0271】I 及び P 映像はそれが復号化されるとき一時デコーダから出力されないことは明らかである。代わりに、I 及び P 映像はフレーム記憶バッファの 1 つに書き込まれ、それらはそれに続く P 又は I 映像が復号化に達したときだけ読み出される。言い換えれば、一時デコーダは 2 つのフレーム記憶バッファから前の映像を読み出して以後の P 又は I 映像を確実にしている。よって、一時デコーダのフレーム記憶バッファを読み出す必要があるとき本発明における空間デコーダは偽りの I 及び P 映像を与えることができる。次にこの偽りの映像は以後のビデオの続きが開始するとき読み出される。

【0272】表 14 に示されるように、映像フレームは番号順に表示される。

【0273】

【表 14】

91

表 示 順	I 1	B e	B 3	P 4	B 5	B 6	P 7	B 8	B 9	I 10
送 信 順	I	P 4	B e	B 3	P 7	B 5	B 6	I 10	B 8	B 9

表 1.4 フレーム記憶

しかしながら、一時デコーダによってメモリ内に記憶される必要があるフレーム数を減少させるためには、フレームは異なる順番で送信される。イントラフレーム（I フレーム）からの分析を開始することは有益である。I フレームは表示されるべき順番で送信される。次の予測したフレーム（P フレーム）、P 4 が送信される。そして、I フレームと P 4 フレームとの間に表示されるべき双方向補間したフレーム（B フレーム）が B e 及び B 3 によって表されるが、送信される。これは、送信した B フレームに前のフレーム（前方予測）又は未来フレーム（後方予測）を参照させるのである。I と P 4 との間に表示されるべき全ての B フレームを送信した後、P 7 フレームが送信される。次に、P 4 フレームと P 7 フレームとの間に表示されるべき全ての B フレーム（すなわち B 5 及び B 6 に対応する）が送信される。そして、次の I フレーム I 10 が送信される。最後に、P 7 と I 10 との間に表示されるべき全ての B フレーム、B 8 及び B 9 フレームが送信される。この送信フレーム順番は、いつも一時デコーダによってメモリに保持されるべき 2 フレームだけを必要とし、差し込む B フレームを表示するために次の P フレーム又は I フレームの送信を待つことをデコーダに要求しない。上記すると共に表 1.4 に示したように、本発明における一時デコーダは MPE G 映像記録を提供するために構成することができる。この映像記録では P 及び I 映像の出力はデータストリーム内の次の P 又は I 映像が一時デコーダによって復号化され始めるまで表示される。

【0274】P 及び I 映像が再配列されるとき所定のトークン、すなわち Picture\_Start、Picture\_Type、及び Temporal\_Reference は、その映像が映像バッファに書き込まれるようにチップに一時的に記憶される。その映像が表示のために読み出されるときそれに記憶されたトークンは得られる。一時デコーダの出力では新たに復号化した P 又は I 映像の D A T A トークンが古い P 又は I 映像の D A T A トークンと置換され、そして、それらはビデオフォーマッタに送られる。一時デコーダからの出力はトークン化されたマクロブロック形式にされ、ブロックからラスタへの変換はない。

【0275】本発明のビデオフォーマッタは 2 つのフレームストア（frames stores）又は映像を記憶する。ビデオフォーマッタにおいては、3 つの映像又はフレームストアが映像を繰り返したり又はスキップするような特徴を達成するために用いられる。ビデオフォーマッタのオフチップ（off-chip）DRAM は 3 つのフレームストアを保持する。ここでの 3 つのフ

レームストアの使用はフレームに、復号化ビデオ及び表示のレートが異なる状態において繰り返し又はスキップのいずれかをさせるのである。

【0276】全ての I、B 及び P フレームはビデオフォーマッタのフレームストアに記憶される。いつも、データが表示されているところからの 1 つのフレームストア、データが書き込まれているところへの 1 つのフレームストアがあり、そして、3 つのフレームストアを有するビデオフォーマッタには他のフレームが第 3 のフレームストアに記憶される。

【0277】従来例は予測、再配列及びブロックからラスタへの変換を全て行い、MPE G は 2 つのフレームストアを有する一時デコーダ及び 2 つのフレームストアを有するビデオフォーマッタ、すなわち全部で 4 つのフレームストアを用いることにより通常処理する。これは本発明において、単に 3 つのフレームストアを用いる機構を共有する 1 つのフレームストアを用いることによって達成される。しかしながら、従来例では 3 つのフレームストアだけでビデオフォーマッタの繰り返し及びスキップ動作を処理することはできない。

【0278】本発明は第 1 のフレームストアに I 映像を記憶し、第 2 のフレームストアに P 映像を記憶する。ブロックからラスタへの変換を行なう必要性から、B フレームは第 3 のフレームストアに以下に示す方法で記憶される。必要とされる外部 DRAM の量を最小にするために機構は引き続き B フレームが同一の第 3 のフレームストアを共有するところに用いられる。

【0279】フレームストアが復号化されるとき、それは第 1 及び第 2 のフレームストアを用いる 2 つの既に復号化された I 又は P フレームを参照する。復号化された B フレームは第 3 のフレームストアに書き込まれる。従来例はフレームストアが完全に満たされる前にラスタを開始させる。そのラスタは、次の B フレームが前のフレームの最初でラスタによって空けられたスペースを埋めるために同一のフレームストアに書き込まれるようにフレームストアが満たされる前に開始される。

【0280】フレームストアの一部分が映像データで埋められ、新たなデータを利用できる記録を保持するために、各フレームストアはセクタの中に分離される。本発明においては、各フレームストアは各々が N セクタからなる 2 つのフィールドストアに先ず分離される。ここで、N はフィールド内のブロック列の数である。フィールド映像として符号化されたフレームは簡単である。各引き続きマクロブロック列は 1 つのフィールドストア内に 2 つのセクタを占める。ライトバック（write back）がフレーム下まで十分に進むと、ラスタは最

初から各セクタを読み出すことを開始する。第1フレームのライトバックが完了すると、次のフレームの開始部分がラスタによって左のスペースに書き込まれる。セクタ各々の状態のチェックは、ラスタが行なわれるセクタが十分に満たされ、ライトバックのために要求される2つのセクタが空であることを確実にする。

【0281】フレーム映像としての符号化されたフレームは更に困難である。フィールド映像とは違って、データのマクロブロック列はそれらがラスタ表示される場合と同一の順番でDRAMに書き込まれない。フィールドストアは並列に書き込まれるが、一方ではそのフィールドは順にラスタ表示される。フィールドストア当たり8セクタを有する映像を考えると、すなわち、フィールドストア0は番号0〜7のセクタからなり、その各々は1ブロック列を含む(すなわち、その映像の幅による奥行き8画素)。フィールドストア1は番号8〜15の8セクタからなり、その各々は1ブロック列を含む(すなわち、その映像の幅による奥行き8画素)。

【0282】第1のマクロブロック列はフィールドストア0にセクタ0内にライトバックされ、フィールドストア1にセクタ1内に書き込まれる。フィールドストアは並列に書き込まれ続ける。ある点にて、ラスタはフィールドストア0からのセクタを表示しており、その点はフィールドストア0のラスタがライトバックに追いつかないように選択されている点である。しかしながら、第2のフレームについては第1のフレームと同一の方法でライトバックすることはできない。そのセクタは異なる順番で書き込まれ、読み出されるので、フレームの開始で空いている同一の2つのセクタを持つことは、書き込み及び読み出しが連続的にできないことを意味する。これは表示を維持し必要なレートでの復号化を維持するために達成されなければならない。

【0283】よって、第2のフレームはラスタによって既に空になったフレームストアのセクタに書き込まなければならない。これはフレームストアを2に分割することによって実行される。従って、第2のフレームに対しては半分のフィールドストアの意味が変わる。セクタ4〜7は第2のフィールドストアの上部となり、セクタ8〜11は第1のフィールドストアの下部となり、すなわち、それらは入れ替えられる。第1のマクロブロック列はセクタ0及び4に書き込まれ、それらが空になると、それに続く列で1及び5、そして2及び6、更に3及び7と書き込みが続く。次の列はセクタ2及び8に書き込まれ、1及び15まで各々書き込まれる。このメモリへの再割り当てはライトバック及びラスタを適切なレートで継続させることにおいて十分である。

【0284】第3の引き続くBフレームが来るならば、ライトバック順は第1のフレームに戻る。共有のBフレームストアにおいて、FRAME(第1)映像では:FIRST映像はセクタ0〜8【第1のマクロブロック列

=2ブロック列]にライトバックされ、よって、1及び9、2及び10、3及び11、...及び15である。

【0285】FIRST映像はセクタ0から、そして、1、2、3、4、5、6、7、8、9、10、11、12、13、14、15の順でラスタ表示される。SECOND(第2)フレームは、セクタ0及び4、そして、1及び5、2及び6、3及び7、8及び12、9及び13、10及び14、11及び15に書き込まれる。

【0286】SECONDフレームは、セクタ0からそして、1、2、3、8、9、10、11、4、5、6、7、12、13、14、15の順でラスタ表示される。本発明においては、第2フレーム、第1マクロブロック列はセクタ0及び1に書き込まれず、それは結局のところラスタによって空にされるべき最初の2つのセクタである。代わりに、それはクリアのためセクタ4を待つ。これは2つの理由のためにされる。第1は、クリアのためにセクタ4を待つことは最も悪い場合の符号化データの状態においてさえ連続的な復号化及び表示を維持するシステムの能力に影響を与えないし、また、実行することは簡単である。第2は、2の乗数でないセクタ数に分割する映像サイズでは、メモリのセクタについて書き込み及び読み出しのシーケンスはたびたび繰り返されない(例えば、NTSC方式はフィールド当たり30セクタを有し、そのシーケンスは58フレーム毎に繰り返す)。これはテスト能力及び回復を困難にする。

【0287】本発明に関する実施である限り、個々のセクタの状態の記録を維持するよりもむしろ、書き込むべき及び読み出すべき次の位置へのポインターをもって、各半分のフィールドストアはFIFOとして効率的に実現される。よって、満たされているか空にされている各FIFOはライトバック及びラスタを各々不可能にする。これは各半分のフィールドストアがFIFOのように1つの方向にだけ書き込まれ読み出されるという知識を利用できる。

【0288】まとめとして、本発明は1フレーム、Pフレーム、B、フレーム及びB2フレームの形でビデオ情報を与え、第1フレームストアに1フレームを記憶させ、第2フレームストアにPフレームを記憶させ、少なくとも2つのメモリ領域に各々分割されている第1及び第2フィールドストアを有する第3フレームストアを与え、第3レジスタにB1フレームを記憶させ、第1又は第2フィールドストア内のメモリ領域の選択した部分からB、フレームを読み出し、B、フレームが読み出されたところのメモリ領域の選択した部分にB2フレームの部分を書き込むことによってビデオ情報を記憶する方法を提供し、それによって減少させた量のメモリをビデオ情報を記憶するために用いることができる。

【0289】以下に見いだされる2つのプログラムは本発明の好ましい実施例に用いられるべき符号を含む。並列ハフマンデコーダについての本発明の詳細な説明

本発明においては、並列ハフマンデコーダブロックは、ハフマン符号化可変長符号 (Huffman code Variable Length Codes: VLCs) 及び固定長符号 (Fixed Length Codes: FLCs) を復号化し、分析マイクロプログラム状態マシン (Microprogrammable state machine: MSM) の制御の元でトークンを介して供給する。

【0290】本発明の実施例はMPEG-1ハフマン符号と同様にMPEG-2も処理する。本発明の実施例の重要な特徴は直列デコーダよりむしろ並列デコーダであるという事実のため高処理能力を維持することができることである。本発明の実施例はハフマン符号を復号するため符号テーブル索引技術を用いる。これはこの動作要求を達成し、実際に不規則又は非標準である第2のMPEG-2変換係数テーブルを処理するためにされる。

【0291】更に、本発明の実施例は外部コントローラの助けなく単一のサイクル内にストリームからのある非常に複雑な成分を復号化することができる特徴を有している。そのような複雑な成分の例はエスケープ符号化係数 (Escape-coded coefficients)、イントラDC値 (Intra-DC values) 及び動きベクトルデルタ (Motion Vector delta) であり、それらの全ては結合したVLC/FLC成分としてストリーム内に与えられる。

【0292】図52を参照すると、並列ハフマンデコーダ300は可変長符号 (VLCs) を処理する。FLCsはデータとFLC長を特定する入力フィールドとを生成するために出力されるセクタ301を使用する迂回機構を要求する。よって、ROM302はFLC復号化の間全く必要とされない。しかしながら、VLCを復号化するため入力、図52に示したように、2つの入力データレジスタ「MSReg」及び「LSReg」に先ず読み込まれる。名称から分かるように、「早い (earlier)」又は最上位データはMSRegに記憶される。そのセクタはROM入力で次のVLCの始めを割り当てのために用いられる。よって、非常に早いVLCを復号するためにセクタはその59ビット入力の最初の28ビットを出力し、それらの最初の16ビットはハフマン符号ROM302に供給される。それに続くVLCに対してセクタはここまで符号化されたビットの合計の計数値に応じた入力を効果的にシフトする。その計数値は復号化されるままに、実行合計に各VLCのサイズを加算することによって保持される。様々なワード幅は、復号することができる最大符号化サイズ (28ビットのMPEG-1エスケープ符号化係数) の結果と、16ビットである最大VLCサイズ (DCT係数テーブル) とある。

【0293】「テーブル選択」入力はMPEGによって要求される様々な異なるハフマン符号テーブル間の選択

のために用いられる。

#### ハフマン符号ROM

VLCs全ての復号化のために用いられる本発明の実行のコアは、図52及び図53に示すように、セクタ/シフト301によってアドレス指定が制御されるROM302である。ROM302はVLCテーブル索引計算を行なう動作を有し、復号化データを生成する索引からデータを得る (index-to-data) 動作が後に続く。

【0294】索引計算は、与えられたデータを生成するハフマン符号を処理するために実行される「ドントケア (don't care: 「1」又は「0」のいずれでも良いビット)」マッチングを有する内容アドレス指定可能メモリ (CAM) の動作として考えることができる。全てのVLC符号テーブルは固定であるので、CAM-ROMは十分であり、これは図54ないし57に示したROM AND-ブレーン (plane: 面) の仕事である。索引生成は (むしろアルゴリズム的に) テーブル索引 (ルックアップ) 方法で行われるので、標準のテーブルを処理する制限はない。

【0295】ROM Or-ブレーンは「索引」(活性化したワードライン) を復号化データ及び符号のサイズ (又は長さ) に変換する。そのデータは復号化出力 (エラーチェックを条件として) を形成し、サイズ情報はセクタを制御する計算を実行させるためにフィードバックされ、よって、それに続くサイクルで次のVLCの復号化を行なうため正しいデータをデコーダROM302に供給する。

【0296】本発明において、ROM302のアドレスは2つのフィールド内にある。大きいフィールドは復号化されるべきビットパターンで、小さいフィールドは調べられるべきハフマン符号を選択する。調べなければならないビットパターンは最も長いVLC符号に対応して長く、16ビットであり、テーブル選択の追加の4ビットがある。よって、合計で20ビットのアドレス空間 (約100万アドレス) があるが、ROM302内に450入力があるだけである。その違いの理由は「ドントケア」ビットの存在のためである。

【0297】VLCsを復号化するためには、AND-ブレーンがVLCビットパターン内の「ドントケア」ビットを復号化することができる必要がある。これは、最大18ビットより短い全てのVLCsが、そのVLCの復号化の部分ではない追加のビットに後を続かっているからである。ワイドアドレスのためAND-ブレーンは事前復号化され (2→4)、ROM302はこの事前復号と共に処理する「ドントケア」を結合しなければならない。更に、完全なMPEG符号テーブルに加えて、ROM302は所定の符号テーブル用に存在する様々なVLCパターンを識別する入力を有する。

#### 【0298】処理能力最大化

97

サイクル毎に1つの復号化項目の出力を維持するために、デコーダ入力制御の上でいくつもの注意がとられている必要があり、特別な処理は「複雑」シンボル（すなわち、それはF L C s又はV L C sではない）のために用いられる必要がある。

【0299】エスケープ符号化係数のピーク処理能力を維持するために、サイクル当たり少なくとも1つの完全な符号を入力しなければならない。要求される最大長はMPEG-1で28ビットであるので、これは（28より大きいその次の実用サイズである）32ビットの入力ワード幅を課している。普通の変換係数も、それらがレベル値の符号を与える1ビットのF L Cがその後続くV L Cからなり、他の複雑なシンボル（例えば、動きベクトル、イントラD C及びエスケープ符号化係数）と同様に処理されるという点において、「複雑」シンボルである。（分離したサイクルで）F L Cがその後続くV L Cとして係数が復号化され、ROM302に符号ビットを復号化させることの代わりの方法がROMに2つの大きなテーブルのサイズを2倍にすることであればピーク処理能力を達成することができない。よって、本発明においては、1つのサイクルが「最終」の要求結果を生成することができるように特別な処理は様々なシンボルのために用いられる。

#### 【0300】F L C及びトークン

F L C処理の基本はF L Cの必要な長さでセレクトを制御すること、ROM302を迂回し正確に選択したF L Cを単に出力することである。よって、単なるF L C sは重要な特別なハードウェアなしでデコーダによって正しく自然に処理される。更に、トークンは処理されないが、デコーダの出力へ直接供給される。

#### 【0301】実施

この章は本発明においてデコーダの実施のいくつかの重要な特徴を述べる。その実施は図52に示すように、カウンタ303及びセレクト301を伴うレジスタの構成と符号ROMを含む。図53の図はコア成分がどのように本発明の主たるハフマン復号化コア部分を実施するために相互接続されているかを示している。レジスタms [31:0]及びls [31:0]は各々MSReg及びLSRegであり、ブロックphselectはセレクトである。カウンタロジック（論理回路）は（様々な他のロジックと共に）ブロックphselectに含まれ、カウンタラッチはcnt [4:0]と表されている。この図上の他のロジックは処理命令、データ及び命令の動き、トークン、（ブロックphselectで行なわれる）更なる「複雑」シンボルの操作を処理する。

【0302】図54に示した図は本発明に応じたハフマン符号ROM302を実現するために用いられたタイプの非常に小さな例のROM設計を示している。このROM302の特有の特徴は、可変長ハフマン符号を復号する方法を実現するために用いられる事前復号及び「ドン

98

トケア」処理がAND-ブレン内に置かれたことである。

【0303】図55、図56及び図57、特に図55を参照すると、「ドントケア」処理を行なうことができるROM AND-ブレンの第1の実施例が示されている。この実施例においては、各アドレスライン（a [3], a [2], a [1], a [0]）はその正方向及び逆方向の両方にAND-ブレンを横切って駆動される。与えられたアドレスライン上の「1」又は「0」を復号化するためトランジスタは従来の方で正又は逆のアドレスラインのいずれかへ接続される。「ドントケア」（xによって示される）を復号化するためにトランジスタはその正又は逆のラインのいずれにも接続されない。

【0304】図56及び図57は復号化ロジック内の最悪の場合の一連のトランジスタの数を減少させるため事前符号化を利用する代替の実施例を示している。それらの例においては、2つのアドレスビットで表すことができる4つの可能な番号の各々について4つのラインのうち1が高レベルに駆動されるような事前符号化において2つのアドレスビットは共に結合される。本発明は2ビットより多いビットが共に結合されるより高いレベルの事前符号化で同様に動作することはこの分野の当業者には明らかである。事前符号化において共にグループとされた2つのアドレスビットが設定値（1又は0、

「ドントケア」ではない）を有するならば、トランジスタは従来の方では適当な事前復号化アドレスラインに接続される。同様に、そのアドレスビットの1が「ドントケア」を有するならば、トランジスタは以前のうに用いられない。しかしながら、アドレスビットの一方が設定値（1又は0）を有する必要がある他方のアドレスビットが「ドントケア」を必要とするならば、事前復号化の2つのアドレスラインのいずれかが活性状態であるとき、選択される0-ブレンを横切って駆動されるワードラインを復号化は要求する。図56に示された実施例において、これは、符号: 001xについての場合に示したように関連する事前復号化アドレスラインの各々上の1つに、並列に2つのトランジスタを設けることによって達成される。図57に示された実施例においては、トランジスタの並列接続を用いることなく要求した復号化が達成される。この場合に、2つの分離した復号化はその両方の選択が必要として行なわれる。それらは、両方の選択が活性ならばワードラインが活性状態となるようなワードラインドライバ内のNORゲートを用いて共に結合される。

【0305】上記の説明は、付随する特徴、目的、及び効果の全てを有する本発明を作成し実行することはこの分野の当業者には可能であるように詳細に本発明の様々な態様の全体コンセプト、システムの実現及び動作を十分に述べていると確信される。しかしながら、本発明及

び本発明の様々な実施例の特に商業的実施に関しての追加の詳細をより深く詳細に理解することを容易にするために、これに続いて更なる記述及び説明が提供される。

【0306】この分野の当業者に対しての説明に役立つ追加の図面は、本発明が機能するような環境の詳細な構成及び動作に更なる理解を与えるためにこの出願と共に含まれる。本発明の上記したパイプラインシステムは、パイプライン処理機器として配置される２線インターフェースによって相互接続された複数のステージ（段）を利用するMPEGビデオ伸張方法及び装置を含むビデオ復号化システムの様々な態様において更なる改善のために長く存在した要求を満たす。制御トークン及びDATAトークンはトークン形式で制御及びデータ両方を運搬する１つの２線インターフェースを介して供給する。トークン復号回路は設けられたステージに適切な制御トークンとして所定のトークンを識別し、パイプラインに介して非認識の制御トークンを供給するために所定のステージに設けられる。再構成処理回路は選択されたステージに設けられ、識別された制御トークンにตอบสนองして識別したDATAトークンを処理するようなステージを再構成する。独特のサポート副システム回路及び処理技術の幅広い変更は、メモリアドレス指定、共通処理ブロックを用いるデータ変換、時間同期、非同期スイングバッファリング、ビデオ情報の記憶、並列ハフマンデコーダ等を含んでシステムを実施するために開示される。

【0307】本発明の特定の形態は示されかつ述べられたが、様々な変更は本発明の精神及び範囲から離れることなくなすことができることは上記のことから明らかである。よって、添付した特許請求の範囲を除いて本発明は限定されるものではない。以下に述べた本発明のシステムのより詳細な説明は、系統化、明確性、及び説明上の便宜のために以下に記した表題に従って記述されている。

概観 . . .  
スタートコード検出器 . . .  
パース . . .  
空間処理 . . .  
予測 . . .  
ディスプレイ回路 . . .  
並列スタートコード検出器 (SCDP) . . .  
入力 FIFO . . .  
入力回路 . . .  
スタートコード . . .  
ビットスタックの除去 . . .  
サーチ・モード . . .  
非整列スタートコード . . .  
オーバーラップスタートコード . . .  
認識されないスタートコード . . .  
拡張及びユーザデータ . . .  
ピクチャ・エンド (PICTURE-END) トークン

の挿入 . . .  
ストップ・アフターピクチャ (Stop After picture) 割り込み . . .  
全廃棄 . . .  
アクセスビット . . .  
SCDPにより認識されるトークン . . .  
SCDPメモリマップ . . .  
実動化 . . .  
コード化データバッファ周辺のデータフロー . . .  
動作の理論 . . .  
不連続性 . . .  
スタートアップ . . .  
実施例 . . .  
ハードウェア . . .  
タイムスタンプ情報のMSMハンドリング . . .  
スタートアップ . . .  
MSMタイムスタンプエラーコード . . .  
30Hzのサポート . . .  
序文 . . .  
状態マシン . . .  
ジャンプ及びコール . . .  
割り込み及びエラー . . .  
ジャンプアドレス . . .  
状態マシン内部の命令 . . .  
状態マシンのテスト . . .  
状態マシン・マイクロ符号マップ . . .  
状態マシン・マイクロ符号語 . . .  
演算コア . . .  
ALU . . .  
シフトブロック . . .  
キャリーブロック . . .  
条件ブロック . . .  
ALUコア . . .  
ALU・マイクロ符号語 . . .  
ALUの使用 . . .  
レジスタ・ファイル . . .  
レジスタ・ファイルアドレス指定 . . .  
レジスタファイル・レジスタタイプ . . .  
レジスタファイル・アドレスマップ . . .  
レジスタファイル・マイクロ符号語 . . .  
トークンポート . . .  
トークンポート・マイクロ符号語 . . .  
マルチプレクサ . . .  
UPIメモリマップ . . .  
序文 . . .  
インターフェース . . .  
機能の記述 . . .  
タイミング要件 . . .  
マイクロプロセッサインターフェースアクセス . . .  
序文 . . .



インターフェース、  
機能の記述、  
誤りを含んで形成せられたトークン、  
ジグザグ・スキャンパス、  
ラスタスキャン順序、  
マイクロプロセッサインターフェースアクセス、  
序文、  
フレーム内の予測画面、  
フレームに基づく予測、  
フィールドに基づく予測(フレーム画面内)、  
デュアルプライム(フレーム画面内)、  
フィールド画面内の予測、  
フィールドに基づく予測、  
16×8MC、  
フィールド画面内デュアルプライム、  
全体的組織、  
水平アップ・サンプラー、  
序文、  
4:3アップ・サンプリング、  
3:2アップ・サンプリング、  
2:1アップ・サンプリング、  
境界効果、  
出力ペル(pel)の数、  
位置信号、  
多重化データ、  
水平位置合わせ、  
アップ・サンプリング比、  
ビデオタイミング発生器、  
序文、  
水平タイミング、  
垂直タイミングPAL、  
垂直タイミングNTSC、  
VTG構造、  
水平マシン、  
垂直マシン、  
ハード結線された比較器のデザイン、  
出力多重化、  
境界生成、  
垂直境界、  
UPI制御、  
出力マルチプレックス、  
概観

この詳細な説明では本発明をチップ全体として扱う。ここで図58を参照すると、そこにはシステムの非常に高水準ブロック図が示されている。以降の節においては、より詳細なブロック図を提供するために各ブロックが拡大されている。

【0308】この記述は回路の種々の機能的ブロックの間のすべてのインターフェースについて正確に文書にしたものである。これにより、各ブロックは、そのブロッ

クが提供することが期待されるインターフェースの完全な知識を用いて設計することが可能になる筈である。図58に示された如く、基本的なシステムの構成部品にはクロック発振器350、スタートコード検出器201、パーサ202、マイクロプロセッサインターフェース(MPI)320、メモリ制御サブ・システム352、空間処理サブ・システム351、予測サブ・システム208及びディスプレイ355が含まれる。図58は更に種々のシステム構成部品の間のインターフェースを図示している。

【0309】スタートコード検出器

図59は本発明によるシステムの回路の他のブロックと相互接続されたスタートコード検出器201(SCD)を示している。SCD201は3つの異なる機能を提供するものと考えることができる。第1に、SCD201は専用のPin又はMPI320からのデータを受け取る入力回路を提供する。第2に、SCD201はデータ内のスタートコードを検出する。第3に、SCDは入力データをコード化データバッファ(CDB)321内で内部的に使用されるフォーマットにアセンブルするのに必要な回路を提供する。

【0310】パーサ

図60は本発明によるパーサ・サブ・システムを図示している。CDB321のためにフォーマットされたデータはアンパックされ、MPI320から命令を受け取るパーサに渡される。その後、データは2線式インターフェースを介してシステムの残りの部分に伝達される。

【0311】空間処理

図61は空間処理回路の構成部品を図示している。これらの構成部品には逆モデラー(Imodel)325、逆ジグザグ器(IZZ)326及び逆量子化器(Iquant)327及び逆離散コサイン変換器(IDCT)328が含まれる。データはImodel325を通り、さらにIZZ326を、次いでIquant327を、さらにまたIDCT328を通過する。

【0312】ディスプレイ回路

本発明のディスプレイ回路が図62に示されている。このシステムは垂直アップ・サンプラー210、水平スケールサブ・システム331、出力マルチプレクサ332及びビデオタイミング発生器333を含んでいる。

並列スタートコード検出器(scdp)

本発明によるスタートコード検出器201は、並列スタートコード検出器、即ち、データを並列に通過させるものである。このシステムは、先に1994年3月24日に出願されたイギリス特許出願第9405914、4号及び1992年6月30日に出願されたEPO出願第92306038、8号、(以下「Broilley」と称する)に開示されたシステムと同様のものである。しかしながら、これら二つのスタートコード検出器の間には幾つかの主要な差異が存在する。第1には、バイト整列が

仮定されている。本発明においては、スタートコードを見つけたためにデータをシフトすることが無いのである。第2に、本発明は主としてMPEGデータに対して動作する。

【0313】MPEG（1及び2）スタートコードはスタートコード・プレフィックス（start\_code\_prefix）として知られるビットストリームにおける唯一のビット（バイトパターン）によって構成されている。このパターンは23個のゼロの後に1が一つ有るものである。このstart\_code\_prefixの直後に続く8ビットはスタートコード・バリュー（start\_code\_value）として知られている。これはスタートコードのタイプを表している。本発明のSCDに到達するスタートコードはバイト整列されていることが必要である。よって、上記のデータは例えば以下のバイト・シーケンスとして特定される。

【0314】

0x00

0x00

0x01

0xb8

これらがグループ・スタートコード（group\_start\_code）である。

【0315】入力Fifo

本発明は250Kバイト/秒のピークデータレートのと  
きにもコード化データバッファがオーバーフローせず、  
入力受け入れ（in\_accept）ピンが引き下げられ\*

スタートコードタイプ	スタートコード値
Picture_start_code	0x00
slice_start_code	0x00から0xaf
reserved	0xb0
reserved	0xb1
user_data_start_code	0xb2
sequence_start_code	0xb3
sequence_error_code	0xb4
extension_start_code	0xb5
reserved	0xb6
sequence_end_code	0xb7
group_start_code	0xb8

表15 スタートコード値

ビットスタップの除去

一つのstart\_code\_prefixに先行する  
いかなるゼロビットもスタップ（詰め物）であり、安全  
に除去し得る。本発明において、スタップの完全なバイ  
トのみ除去される。

【0319】例えば、以下に示すバイト・シーケンスに  
おいては13のスタップ・ビットが有り、その内の8個  
のみが実際に除去される。

0x20 //5スタップ・ビット

\* れることがないように設計されている。従って、入力  
fifoの長さを計算するためには、1）シングラッパ  
フのための最悪の場合の待ち時間と、2）SCDを通し  
た最悪の場合のデータの拡大について知る必要がある。

【0316】本発明によれば、コード化データクロック  
レートに到達する入力データに対して、SCDPはスタ  
ートコード毎に二つのストールを生成する（データス  
トリームから3バイトが除かれている）。

入力回路

本発明の入力回路はBrollによって開示されたも  
のと全く同一の方法で動作する。しかしながら、それら  
の二つの回路の間には2、3の重大な差異が存在する。

第1に、upiはトークンの有効なエンド（それがセッ  
トされないかも知れないため）まで待つように作られて  
いない。その代わりに、信号イン・トークン（in\_t  
oken）がローになるまで待つように作られる。第2  
に、バイトモードに入る際のDATAレヘッダーの生成  
は、いくつかのバイトモードデータがあることに依存す  
る。

【0317】スタートコード

本発明においては、MPEGスタートコードはSCDに  
よって認識されてトークンに変換される。それらを表1  
5に示す。

【0318】

【表15】

40 0x00 //8スタップ・ビット

0x00

0x00

0x01 //

start\_code\_prefix

サーチ・モード

本発明によるサーチ・モード（search\_mod  
e）は以下の表16に示されている。

【0320】

【表16】

サーチ・モード	動作
0	通常動作
1	picture_startまたはそれ以上のサーチ
2	group_startまたはそれ以上のサーチ
3	sequence_startまたはそれ以上のサーチ

表16 サーチ・モード

ゼロ以外の何れのサーチ・モードではスタートコードの所望のクラスが見つかるまで到着するデータの全てが廃棄される。その時点で、サーチ・モードがゼロにリセットされ、スタートコード・サーチ (start\_code\_search) が割り込みが発生する。新しい制御ビットであるストップ・オンサーチ (stop\_on\_search) がSCDが割り込み (この割り込みはまた通常の方法でマスクされるけれども、停止は必須ではない) を発生した後に実際に停止するか否かを判別する。

【0321】本発明においては、SCDがFLUSHトーンを受け取った場合にもまたsearch\_modeがゼロにセットされる。しかしながら、FLUSHトーンがdiscard\_allで終了する場合にはsearch\_modeが完全にリセットされる。即ち、FLUSHトーン及びdiscard\_allの組み合わせによってサーチ・モードがリセットされる。

#### 【0322】非整列スタートコード

一つ以上のゼロバイトの連続の後に0x01が続けばスタートコードである。更に、23個以上のゼロの連きの後に1が続かなければ非整列スタートコードである。バイト整列された環境ではこのことは以下のように解釈される。もし、ビットスタフを除去した後に0x01を受けとらない場合にはスタートコード非整列である。なお、この宣言は非整列スタートコードの有るもの(スタッフ)が1バイト以下である場合に関係している)は見落としていることに注意すべきである。

【0323】本発明のSCDPは、データシートにおいて非整列スタートコードのどのクラスが検出されるかについて記述する努力をするよりはむしろ、それらを無視している。換言すれば、スタッフはまだ除去されているのである。

#### オーバーラップスタートコード

スタートコードの「値」の部分が以降のスタートコードの「prefix」の部分形成するようにすることが可能である。これは一般的に以下の二つの理由により生起する。1) 規格によればシステムレベルのスタートコードはストリーム内のどの位置に生起しても良く、ビデオレベルのスタートコードの中に直接生起することも含まれる。2) エラー。誤りと思われるスタートコードをすべて最後まで除去し、それによりエラー復旧の可能性が高まる。

【0324】バイト整列環境においては、本発明によれば、オーバーラッピングスタートが生起するただ一つの方法はピクチャースタート (picture\_start

t、値=0x00) が他のスタートコードの部分となる場合のみである。このやり方で、picture\_startはデータから除去され、第2のスタートコードがデコードされる。もし、次にこれがオーバーラップしていれば、非オーバーラッピングスタートコードが検出されるまで同一の手続きが行われる。

#### 【0325】認識されないスタートコード

本発明において予約値 (0xb0、0xb1、0xb6)、全部のシステムスタートコード (0xb9から0xff) 及びシーケンス・エラーコード (0xb4) はそれぞれ認識されないスタートコードである。認識されないスタートコードを除去した後は、SCDは次の有効なスタートコードが見つかるまで全ての入力データを廃棄する。またSCDは認識されないスタート (unrecognized\_start) エラーレジスタをセットし、認識されないスタート (unrecognized\_start) マスクに依存して割り込みを生成する。

#### 【0326】

##### 拡張及びユーザーデータ

本発明において二つのコンフィギュレーションビットが使用される。

##### 1) Discard\_user (又は無し)

##### 2) Discard\_extn (MPEG2メイン・プロファイル、メインレベル以上)

これらの二つのコンフィギュレーションビットは1にリセットされる。

【0327】MPEG2拡張スタートコードは異なっている。拡張スタート・バリュー (extension\_start\_value) に続く4つのビットは拡張スタートコード識別子 (extension\_start\_code\_identifier) であり、SCDによってデコードされねばならない。4つの新しいトークンがこれらをフラグするために生成されている。許容された拡張コード識別子 (extension\_start\_code\_identifiers) 及びそれらのトークンを表17に示す。しかしながら、予約された拡張スタートコード識別子 (extension\_start\_code\_identifier) は認識されない。認識されない拡張スタートコード (extension\_start\_codes) は廃棄されるか (Discard\_extnに依存)、又は (ID) extension\_dataトークンによって置換される。

#### 【0328】

【表17】

拡張スタート コード識別子	名 称	新トークン	ヘッド
0000	予約		
0001	シーケンス拡張ID	SEQUENCE_EXTN	0xe8
0010	シーケンスディスプレイ拡張ID	SEQUENCE_DISPLAY_EXTN	0xe9
0011	quantマトリクス拡張ID	QUANT_MATRIX_EXTN	0xea
0100	予約		
0010	サーチスタート拡張ID		
0110	予約		
0111	ピクチャーサイズ拡張ID		
1000	サーチフィールド拡張ID	PICTURE_CODING_EXTN	0xeb
1001	ピクチャー空間サーチ拡張ID		
1010	ピクチャー時間サーチ拡張ID		
1011	予約		

表17. MPEG2拡張スタートコード識別子

ピクチャー・エンド (PICTURE\_END) トークンの挿入

現行の標準 (MPEG1、2、JPEG、又はH26

1) はいずれも現在の画面を終了する方法について規定 20  
していない。

【0329】しかしながら、本発明においては、SCD 201がイン・ピクチャー (in\_picture) と呼ばれる状態を保持する。この状態は PICTURE\_START トークンが SCD 201によって出力される際にはいつでもセットされる。構文上において picture\_start (又は FLUSH トークン) より高いそれ以降のスタートコードは PICTURE\_END トークンの生成を起こす。PICTURE\_END トークンは生成されて新しいスタートコードに関係するいかなる 30 トークンより前に出力される。状態 in\_picture は PICTURE\_END トークンが SCD 201を離れる際にリセットされる。もし SCD 201が入力データストリームでトークンを受け取った場合でも、PICTURE\_END トークンを受け取る場合も含めて動作は論理的に同一である。要約すれば、PICTURE\_END を生成させる可能性のあるスタートコード (及びトークン) は、本発明によれば以下の通りである。

【0330】

picture\_start\_code 又は トークン group\_start\_code 又は トークン sequence\_start\_code 又は トークン sequence\_end\_code 又は トークン FLUSH トークン

ストップ・アフタ・ピクチャー割り込み

ストップ・アフタ・ピクチャー (sap) の特徴は、現在のシーケンスを終了させる簡潔な方法、例えばチャンネル変更を容易するための本発明の機能である。この機能 50 をできるだけ自動的に外部のリアルタイムソフトウェア

を必要とすることなく達成することが必要である。

【0331】sap 制御ビットはフラグ・ピクチャー・エンド (flag\_picture\_end) と称する。flag\_picture\_end、マスク及びエラービットに加えて以下の二つの制御ビットが存在する。

1) after\_picture\_stop: 割り込みを生成して後に SCD が停止するか否かを決定する。

【0332】2) after\_picture\_discard: このビットは SCD が flag\_picture\_end 割り込みを生成した後に discard\_all モードに自動的に移行するか否かを決定する。この方法により、discard\_all モードはどのイベントがそれを呼び出したかについて知る必要がなくなり、discard\_all モードをそのままにして瞬時にかつ簡潔にサーチ・モードに移行することが可能になる。

【0333】本発明に依れば、PICTURE\_END トークンが SCD により出力される際には常に flag\_picture\_end ビットが何か動作をなすべきか否かを決定する。もし flag\_picture\_end がセットされていれば、PICTURE\_END の後に FLUSH が生成され、イベントが生成される。割り込みは flag\_picture\_end\_mask に依存し、(割り込みの場合には) ストップは after\_picture\_stop に依存する。

【0334】一例として、チャンネル変更のためのイベントのシーケンスは以下の如くである。

1) after\_picture\_stop = 0 及び after\_picture\_discard = 1 で flag\_picture\_end を セット。

【0335】2) flag\_picture\_end\_event に応答。

a) サーチ・モードをシーケンスにセット (一例)。

b) リチューン等。

3) discard\_allをFLUSH又はs/wリセット。

4) SCDPが次のシーケンスの開始をサーチ。

【0336】全廃棄(discard\_all)

R/W制御ビットのdiscard\_allが本発明によるSCDPをしてFLUSHトークンまで、それを含めて全ての入力を廃棄せしめる。このビットはFLUSHトークンによって自動的にリセットされ、flag\_picture\_end関数によってセットされ得る。\*

\*【0337】SCDPによって認識されるトークン  
 大概の本発明のSCDPの主要な機能は実際のトークン生成に關しているが、幾つかのトークンはコード化データポートに供給され(又は入力回路を介し)たときにSCDPによりデコードされ、実行される。これらのトークンは表18に示され、定義されている。

【0338】

【表18】

トークン	ヘッダー	動作	解説
FLUSH	0x17	scdpフラッシュ	これらのトークンはPICTURE_ENDを生成し得る。この場合に、それらのトークンはIn_pictureをリセットし、flag_picture_endイベントを生成させるとともにFLUSHを生成し得る
PICTURE_START	0x12	in_picture セット	
PICTURE_END	0x16	in_picture リセット	
GROUP_START	0x11		
SEQUENCE_START	0x10		
SEQUENCE_END	0x14		
DATA	0x04等	データ内でスタートコードをサーチする	
Other	—	認識されないトークンは変更無しにSCDP内を通わせる。	

表18 認識される入力トークン

Scdpメモリーマップ

※【0339】

本発明のSCDPのための種々のレジスタ及びそれらの

【表19】

関連するアドレスが表19に記述されている。 ※

レジスタ名	ビット	bit	解説	アドレス
scdp_access	0			0x0
scdp_access	[0]	0	アクセスビット	
scdipe_od [7:0]				0x1
CD0 [7:0]	[7:0]	1	sp12-12データポート	
scdipe_od1 [7:0]				
coded_busy	[7]	1	リード・オンリー	
enable_coded	[6]	0		
coded_extn	[7]		リード・オンリー	
scdp_ctl0 [7:0]	000		0x03	
discard_extn	[5]	1		
discard_user	[4]	1		
discard_all	[3]	0	FLUSHによるリセット	
flag_picture_end	[2]	0	イベント可能化	
after_picture_stop	[1]	0	イベント可能化は有効	
after_picture_discard	[0]	0	イベント可能化は有効	
scdp_ctl1 [7:0]		0		0x4
stop_after_search	[2]	0	イベント可能化は有効	
start_code_search[2:0]	[1:0]	0		
scdp_event [7:0]		0		0x5
end_search_event	[0]	0		
unrecognized_start_error [1]	0			
flag_end_of_picture_event	[0]	0		
scdp_mask [7:0]		0		0x6
end_search_mask	[2]	0		
unrecognized_start_mask	[1]	0		
flag_end_of_picture_mask	[0]	0		

表19 並列スタートコード検出器メモリーマップ

コード化データバッファ回りのデータフロー

【0340】

本発明は以下の利点を提供する。

50 1) バッファを回転(swing)させる方法。

2) バイトを奇数のビットにパックする必要を回避する方法。

3) (長くなる可能性の有る) SCDのバス幅を8ビットまで減少させる。

4) SCDが自身の32ビットデータへのパッキングを行う。大きなバスを回避するために、SCDのこのビットはDRAM-IFの内部に位置している。本発明において、これをscddbінと称する。このモジュールは全てのDATAを32ビット語にパックし、非データトークンの間は推算する。

【0341】5) スイングバッファがそれ自身の計数及びスイングを行う。バッファはPICTURE\_END又はFLUSHトークン(又は信号)に応じてscddbінからの信号fill\_and\_swingにに応じてフラッシュする。

6) ハフマンデコーダの前に位置するアンパックモジュールscdboutが出力スイングバッファにより提供されるbuffer\_start信号を受け取るまでFLUSH又はPICTURE\_ENDに続く全てのデータを検出する。

序文

この節では本発明によるタイムスタンプ情報の扱いが定義される。

#### 【0342】動作の理論

MPEG-2ビデオ及びオーディオにおいてはデータはMPEG-2システムストリームによって運ばれた情報を使用して同期が取られる。同期、クロック基準及びタイムスタンプを扱う情報には本質的に二つのタイプが有る。クロック基準はデコーダに「現在」時間を表すためにどのような数が用いられているかについて知らせるために用いられる。これは、規則的な間隔でインクリメントされるカウンタを初期化するために用いられ、それによりデコーダは常に現在時刻が何時であるかについての概念を持つことができる。

【0343】(典型的な例としてビデオ及びオーディオ)を形成するために用いられるデータのストリームの各々についてタイムスタンプが付加されている。ビデオの場合、タイムスタンプは画面に関係づけられ、デコーダに対して「何時」(クロック基準により初期化されたカウンタにより定義される)画面を表示すべきかを知ら

せる。【0344】しかしながら、MPEGにおける全ての物と同様に、状況はこれよりずっと複雑である。クロック基準には二つのタイプが存在する。プログラムクロック基準(PCR)と、システムクロック(SCR)である。クロックは90KHzの解像度までの情報を有しており、他方おクロックは解像度を27MHzまで拡大する追加の情報を有している。クロック基準は「時間」がランダムアクセス又はチャンネル変更の後に再度初期化可能であるようにデータストリーム内にかなりしばしば

含まれている。

【0345】タイムスタンプにもまた二つのタイプがある。プレゼンテーションタイムスタンプ(PTS)およびデコーダタイムスタンプ(DTS)である。これらは、並べ替えられねばならないI-ピクチャ及びP-ピクチャについての異なっている(B-ピクチャについては同じである)。DTSは画面をいつデコードすべきかを示す一方、PTSは画面をいつ表示すべきかを示す。単純な、2-3プルダウン効果の無いフレーム画面の場合にはI-ピクチャ又はP-ピクチャのDTS及びPTS間の差異はその画面フレーム期間に続くB-ピクチャの数より1多くなる。

【0346】理解しておくべき重要な複雑さはDTS及びPTS画面を瞬時にデコードすることが可能なデコーダの仮想的なモデルに関するものであるということである。実際のデコーダはいずれもこのようなことはできず、デコーダが画面を表示すべき理論的な時間を修正するためには多くのステップ必要とする(タイムスタンプ及びクロック基準によって定義される)。このような変更例はデコーダのアーキテクチャーの詳細に依存する。明らかに、ビデオデコーダにより生じたいかなる遅延も、音声デコーダ内の等価な遅延によって整合されねばならない。

不連続性

「時間」の概念での不連続性が生じる可能性が有る。例えば、編集されたビットストリームにおいて各編集点は不連続な時間を有することになる。同様の状態はチャンネル変更の際にも生起する。一つの時間管理状態でエンコードされたタイムスタンプを他の管理状態からのクロック基準によって定義される「時間」に対して用いることは明らかに不正確な結果に到るため、注意しねばならない。

スタートアップ

スタートアップ(又はチャンネル変更)においては、正確にデコードを開始するために二つの潜在的に適合する要件が存在するため、特別な問題が生じる。ビデオについて考慮すると、システムバッファに続くI-ピクチャからデコードを始めなければならぬ(これは全ての場合に正しくは無いが、概ね正確である)けれども、システムについて考慮すると最初にデコードされた画面をタイムスタンプを保持している必要がある。しかしながらすべての画面がタイムスタンプを保持していなければならないという要件は無く、もしI-ピクチャであって、かつタイムスタンプを有する画面を探索しようとするば永遠に待ち続けなければならないかも知れない。

【0347】一つのI-ピクチャについて、それに行先するタイムスタンプを有する画面からタイムスタンプがどうなっていたであろうかについて計算することを望むつくかも知れない。残念ながらこの方法は、それらの間

にある画面がフィールド画面であるかフレーム画面であるか(そしてrepeat\_first\_fieldがセットされているか否か)を判別するためにその中間の画面を部分的にデコードしなければならないため、実行することは非常に難しい。このことはデータがコード化データバッファ内を通過するとともにハフマンデコーダにより廃棄される必要がある。

#### 【0348】実施例

図63はタイムスタンプの管理を実現するための第1の実施例を示している。クロック基準253は本発明のシステム分離器254によってデコードされて、90KHzでインクリメントされる時間を表すカウンタ255に送られる。それらはまた、ビデオデコーダ270内に位置する第2のカウンタ258にもロードされる。

【0349】タイムスタンプはビデオバッファ271を通過し、ビデオデータと同一の量だけ遅延される。これらは次に時間の局所コピーと比較されて画面が早すぎるか遅すぎるかが判別される。本発明による他の実施例が図64に示されている。この例はクロック基準253がビデオデコーダ270に渡される必要をなくしたものである。このことはビデオデコーダ270及びシステムデコーダ256の両方の内に維持される第2のカウンタ「vid\_time」272、273を用いることによって行われる。それらは電源オンによりリセットされ、その後フリーランする。この実施例においては二つのカウンタが調和して動作することが要求されるため、それらの調和が与えられることが無いようにする必要がある。このためにはシステム分離器(demux)内のカウンタのキャリア出力をビデオデコーダ(図示の如く)内のカウンタをリセットするために用いることで達成できる。

【0350】この実施例の他の利点としては数の全33ビットを扱う必要がないことが挙げられる。理想を言えばビデオデコーダ270における16ビットの扱いが可能にするためにカウンタを16ビットに限定することが必要である。このようにすると、90KHzの解像度では不十分な数の範囲のみ(2/3秒のみ)しか表せないように見えるけれども、VTGはフリーランニングする(又はデコードされるMPEGストリームとは関係の無い何かにロックしている)ため、ビデオデコーダにおいては時間制御はいずれにしてもフィールド時間のみ正確であれば良く、その様な高精度は必要としない。

【0351】その結果、デコーダに入るタイムスタンプの下位の数ビットは廃棄される可能性があると思われる。本発明においては、4つのビットが廃棄される。このことはビデオデコーダ20のビット数の16ビットを使用することを意味している。解像度は、こうして5625Hzとなり、11.65秒の時間差を表すことができる。

【0352】従って、PALフィールドは5625Hz

のクロックの112.5回となる。NTSCフィールド93.84回となる。よって、タイミングの計算はまだフィールド時間の約1%精度で行うことが可能であり、本発明にとって充分である。

#### ハードウェア

図65は本発明によるハードウェアを示している。Block 11に開示されたモジュールに加えて二つのモジュールがある。第1のものはスタートコード検出器201の直後に付加され、トークンの生成を担当する。TIME\_STAMPトークンがPICTURE\_STARTトークンの直前に生起する。MPEGシステムストリームにおいて、タイムスタンプはパケットヘッダー内に保持され、データのパケット内の最初の画面を示している。パケットはビデオデータと整理していないため、一般に、タイムスタンプが示している画面の開始の前に前の画面の終端が存在することになる。

【0353】タイムスタンプ情報マイクロプロセッサインターフェースを介するか又はトークンを用いて本発明のシステムに供給することができる。どちらの場合にも、タイムスタンプデータ(16ビット)がレジスタに格納される。タイムスタンプ情報がレジスタ内にあることを示すためにフラグがセットされる。もしTIME\_STAMPトークンを用いてデータが供給された場合にはそのトークンはトークンのストリームから除去される。

【0354】PICTURE\_STARTトークンに遭遇したときには、レジスタの状態を示すフラグが調べられる。もしクリアであれば、何も動作は行われず、PICTURE\_STARTトークン及び以降の全てのデータは影響されない。いっぽう、もしフラグが有効なタイムスタンプ情報がレジスタ内に存在することを示していれば、PICTURE\_STARTトークンの前にTIME\_STAMPトークンが生成される。その後、フラグはクリアされ、次に生起するタイムスタンプについて使用可能となる。

【0355】第2のハードウェアモジュールはマイクロプログラム可能な状態マシン218に関するものである。これは、単に27MHzデコーダクロックからクロックを受ける一連のカウンタである。第1のカウンタはクロックを4800(図に示された480.8のオプションについては後に記述する)で分周するプリスケールである。4800は300(27MHz/90KHz)を16倍した数である。第2のカウンタは時間カウンタであり、プリスケール27MHzがクロックを出力する度にインクリメントされ、reset\_timeピンによってリセットされる。

【0356】この部分のカウンタはその他の部分で使用される弱フィールド・バックラッチに比べて粒子汚染に対して耐性が高い完全クロックフィールド・バックフリップフロップ(同期の)によって実現される筈である。

(これはBrianの時間カウンタがシステムデコーダ内のカウンタと調和しない虞れがあるからである。) マイクロプログラマブル状態マシン218は時間カウンタによって示される現在時刻を読み出し、それをTIME\_STAMPトークンによって供給される値と比較することが可能である。従って、それが画面をデコードし\*

\* ているべき時間に比較して早いか遅いかを確認することが可能である。

【0357】タイムスタンプに関するSCD201において用いられるレジスタは表20に示されている。

【0358】

【表20】

レジスタ名	大きさ/ 方向	94137-1	記 述
ts_low	8/rw	-	タイムスタンプ値の下位8ビットこのレジスタはスレープとされ、先に書き込まれた値 (TIME_STAMPトークンの一部となる) に影響を与えることこのレジスタに新しい値が書き込むことができる。 このレジスタへの書き込みはスレープレジスタからリード・バック (読み戻し) している間はマスターレジスタに影響を与える。ts_validを用いてマスターからスレープへの転送が行われるまではts_lowに書き込まれた値はリード・バックできない。
ts_high	8/rw	-	8つのタイムスタンプ値の上位8ビット ts_lowと同様にスレープとされる。
ts_valid	1/rw	0	このビットはts_low及びts_highのマスター・スレープ転送を制御する。 ts_low 及びts_highに値が書き込まれたときに、マイクロプロセッサは値1をこのビットに書き込む必要がある。その後、マイクロプロセッサは値1をリード・バックするまでこのビットをポーリングする必要がある。この時点において、ts_low及びts_highに書き込まれた値はスレープレジスタに転送されて (従ってリード・バック可能になり) あり、ts_validが1にセットされる。 マイクロプロセッサは次のアクセスに対して準備するために値ゼロを書き込む必要がある。



ts_waiting	1/rw	0	<div data-bbox="727 126 751 143">118</div> <p>ゼロにセットされたとき、レジスタts_low及びts_highは有効なタイムスタンプ情報を含まない。</p> <p>1にセットされたとき、レジスタts_low及びts_highは有効なタイムスタンプ情報を含む。TIME_STAMPトークンが次のPICTURE_STARTトークンの前に生成され、ts_waitingがゼロになる。</p> <p>このビットは、前回のタイムスタンプ値がマスター・スレーブ転送により上書きされる前に使用されたことを確認するために、1をts_validに書き込む前にそれがゼロであることを確認するため、ポーリングされることが必要である。</p>
------------	------	---	---

表20 タイムスタンプ「SCD」レジスタ

MSMによるタイムスタンプ情報の扱い

この節においては、本発明によるMSM218の、TIME\_STAMPトークンを受け取った際の機能の詳細について記述する。

【0359】はじめに、16ビットの符号付きタイムスタンプ補正がTIME\_STAMPトークンによって担持されるタイムスタンプに付加される。この補正はチップリセットの際にMSM218によりリセットされ、もし何も動作が行われない場合にはタイムスタンプは変化されない。制御用マイクロプロセッサは、しかしながら、タイムスタンプを修正するためにこのレジスタにどのような値も書き込むことができ、それにより、ビデオ及び音声デコーダを通した差分による遅延を補償する。

【0360】次に、補正されたタイムスタンプが現在時刻から減算される。この結果の符号がエラーの方向を与える（そしてもし有れば、MSM218によって生成されたエラーコードを判別する）。次いで差の絶対値がとられ、その結果がフレーム時間と比較される。もしその結果がフレーム時間より小であれば、何も動作は行われない。前に述べたように、VTGがフリーランニングするため、時間は公称時間からフレーム時間をプラス・マイナスした精度でのみ制御可能である。

【0361】本発明において、もしエラーがフレーム時間を越えた場合には、何等かの補正が行われなければならない。MSM218は復号が早すぎる場合には適切な時間になるまで単に復号を遅延させることができるため、それ自身で状態を補正することができる。しかしながら復号が意図された時間より遅い場合には、コード化データバッファの出力において画面を信頼性をもって産

棄することはできないため、これはもっと複雑になる。本質的には、シーケンスの復号は壊れており、状態を補正する最も信頼できる方法は、ランダムアクセス又はチャンネル変更の場合と同様の方法で復号処理を再スタートすることである。この手続きを容易化するために、MSM218の制御レジスタFLUSHトークンに連通するまで全てのデータを廃棄するようにプログラムしても良い。

#### スタートアップ

もし本発明によるMSM218が、タイムスタンプをそれがスタートアップ状態として認識する時間に受け取った場合には（例えば、リセットの後、SEQUENCE\_ENDトークン又はFLUSHトークンの続きでまだPICTURE\_STARTの前のとき）、MSM218の動作は修正される。もしタイムスタンプが現在時刻より前に復号が行われていなければならないことを示していれば、上に詳述した方法と同一の方法で状態が扱われる。しかしながら、もしタイムスタンプが現在時刻より後に復号が行われるべきことを示している場合（これはスタートアップの通常の状態である）には、もしエラーが1フレーム時間より短くてもデコーダは正確な時間まで待機する。この方法で、公称復号時間を正確な時間に対してできるだけ正確に設定することができる。その後の画面はそれらの公称時間の前あるいは後に、エラー状態がトリガーされることなしに1フレーム時間までデコードされる。

【0362】加えて、本発明においてはエラー「ERR\_TOO\_EARLY」はdisably\_too\_earlyの設定に拘らずスタートアップの間には生成されない（このエラーでは、復号が早いことが期待される

ため)。

MSMタイムスタンプエラーコード

タイムスタンプの処理の結果として、二つのエラーの内の一つが生成され得る。復号がタイムスタンプにより示される時間より早く行われた場合にはERR\_TOO\_EARLYが生成される。

【0363】タイムスタンプにより示される時間より遅く復号が行われた場合にはERR\_TOO\_LATE。\*

\* ERR\_TOO\_EARLYは抑圧することができるが、ERR\_TOO\_LATEは全てのエラーがマスクされない限り常に生成される。表21は本発明によれるマイクロプログラマブル状態マシンに係る種々のタイムスタンプレジスタを記述したものである。

【0364】

【表21】

レジスタ名	サイズ	リセット状態	記述
ts_correction	16/rw	—	各タイムスタンプにはそれが使用される前に補正が加えられる。
frame_time	16/rw	226 or 188	画面のデコードのタイミングの許容範囲を表している。リセット状態がPAL/NTSCピンにより決定される。
time	16/ro	ゼロ	リセット又はtime_resetの何れかによりリセットされる。現在時間の値。
manual_startup	1/rw	ゼロ	1にセットされたときに、復号が早すぎることを示すエラー「ERR_TOO_EARLY」は抑圧され、MSMは単に正確な状態を待つ。 1にセットされたとき、スタートアップはdecode_disableを使用して手動で行われる。この場合MSMにおけるSEQUENCE_END及びFLUSHトークンがdecode_disableを1にセットさせる。ゼロにセットされたとき、スタートアップはタイムスタンプ管理ハードウェアを用いて行われる。Decode-disableは自動的に1にセットされることは無い。
decode_disable	1/rw	ゼロ	ゼロにセットされたとき、復号は通常の方法で進行する。

121

122

disable_too_early	1/rv	ゼロ	各画面のスタートにおいて、MSMはdecode_disableの状態をチェックし、もしそれが1にセットされていれば先に進まない。 もし手動のスタートアップが行われるべき場合には(即ち、タイムスタンプ管理ハードウェアを用いない場合)にはこのビットはmanual-startupが1にセットされるのと同時に1にセットされねばならないことに注意されたい。
NTSC_30	1/rv	ゼロ	1にセットされたとき、プリスケールは4800ではなく4804.8で分周する。30Hzフレームの速度でデコードするときに自動的に設定される。
discard_if_late	1/rv	ゼロ	これは「ERR_TOO_LATE」が生成されない限り何も効果を持たない(又はエラーがマスクされていなかった場合に生成される)。もしそれが1にセットされていればdiscard_untilにより示される条件が得られるまでデータは廃棄される。
discard_until	2/rv	0	タイムスタンプによりトリガーされる廃棄が終了する条件を示す。 0-PUSH 1-SEQUENCE_START 2-GROUP_START 3-Next Picture 注1- 1画面の廃棄は、その画面がフィールド画面であるときには、ダミーフィールドを生成して交替する上/下フィールド構造を保存することにより直に取り消すことができる。その結果、もしdiscard_untilが「Next Picture」にセットされるがダミーフィールドが生成される場合には、更に1画面が廃棄される。

表21 タイムスタンプ「MSM」レジスタ

## 30Hzのサポート

本発明は30Hzフレームレートは正確にはサポートしない。しかしながら当業者には、本発明はもしクロック生成回路が適切に変更されれば30Hzデータをデコードすることができることが理解される。この場合には、システムは27.027MHzクロックに対応するために、それを300.3で分の1に分周して90kHzクロック生成しなければならない。本発明においてはこの

40 値を16を係数として拡大しているため、クロックを4804.8で分周する必要がある。

## 【0365】序文

この節では本発明によるマイクロコーディング可能な状態マシン(MSM)の詳細について記述する。MSMを構成する目的は、わずかな補正でVLCデコーダ及びアドレス発生器などの多くに応用することが可能なマシンを製造することである。

【0366】本発明のMSMは広い範囲の特徴をサポートする汎用的なものである。しかしながらMSMの基礎を成す構造はモジュラー形式であり、柔軟に構成するこ

とが可能である。よって、当業者には本発明は種々の応用を使用することが可能であることが理解される。図6に示された如く、システムのデザインは二つの区画に分割される。第1の区画は状態マシン218である。既にBrollyの出願に開示され、ここに引用として含める2線式インターフェースによって制御されるデータ処理パイプラインに渡される命令を生成する。第2の区画はALU222及びそれに連合するレジスタファイル221から成る演算コア219である。この演算コア219はデータ処理パイプラインの一部である。これは二つの2線式インターフェース注1による制御に従ってデータ及び命令を受け入れる。それは、一つの2線式インターフェースによる制御に従ってその出力にデータを発生する。これらの二つの構成部品を組み合わせることによって完全なマイクロ符号語(マイクロ・コード・ワード)を定義することが可能になる。

注1 状態マシンはまたアップ・ストリームブロックを制御する場合には、これらの二つの2線式インターフェースは組合せられる。

#### 【0367】状態マシン

本発明による状態マシン218は演算コア219に命令を与える。それはまた命令の進行に従って制御自身に命令を与える。演算コア219に渡された命令のアドレスはプログラムカウンタに保持される。プログラムカウンタ

\*タは0x00にリセットし、そのアドレス内を連続的に進む。しかしながら「ジャンプ」または「call」命令及び/又は「割り込み/エラー」イベントによりプログラムカウンタは再ロード可能であり、従って命令の実行の順序を変更可能である。

【0368】本発明においては状態マシン218は4096の命令まで受け入れることが可能である。しかしながら、これ以外の量の命令を使用することも可能であり、これは限界として作用することを意図するものではないことは当業者に理解される。

#### ジャンプ及びコール

この実動化例において、全ての命令は条件ジャンプ命令である。条件は全ての命令について評価され、ジャンプすべきか(即ち、プログラムカウンタを再ロードすべきか)否かが決定される。無条件ジャンプ又はジャンプ無しには二つの条件「真」及び「偽」が設けられている。残りの条件(合計16)はステータスバス上のテストに基づいている。もし条件が「真」又は「偽」でなければ、状態マシン218は演算コア219が命令を実行し終えて条件に対するテストのためにステータスバスを状態マシンに戻すまで待つ。これらの条件は以下の表22に示されている。

#### 【0369】

#### 【表22】

コード	条件
0001	F 偽-ジャンプしない
0010	C キャリーセット
0011	NC キャリークリア
0100	Z ゼロ
0101	NZ 非ゼロ
0110	AN ALUの結果が負
0111	AN ALUの結果が正
1000	F 偽- 予備条件
1001	F
1010	LT (S ^ V) [-]は <J> を示す
1011	GE (S ^ V) [-]は >J> を示す
1100	I インデックスレジスタインクリメントが終点を通過。
1101	NI インデックスレジスタインクリメントが終点を通過せず。
1110	V オーバフロー
1111	NE 拡張ビットがロー

表22 状態マシンの条件

もしコールビットがセットされた命令でジャンプが行われたときには、ジャンプが行われていない次のアドレスがリターンアドレスとして格納される。よって、これがルーチン呼出しの機構を形成する。ルーチンから格納されたアドレスに戻るためには、アドレス0x001にコールが行われる。コールは1コールの深さだけサポートされる。即ち、ただ一つのリターンアドレスのみが格納され得る。しかしながら、コールからのコールは、誤ってはいけるけれども、ハードウェア内ではチェックされな

い。

#### 割り込み及びエラー

本発明において、もし割り込み/エラー線が高いであるとサンプルされれば、割り込み/エラーアドレス(アドレス0x001)への無条件ジャンプが行われる。割り込み/エラーが無かった場合に取込まれてはいたはずの次のアドレスが格納される。割り込み/エラールーチンから戻るためには、割り込みアドレス(0x001)へのジャンプが行われる。

【0370】本発明による状態マシン218は、割り込み又はエラールーチンとしての実行のためにハード結線されている。その違いは割り込みルーチンは実行中は他の割り込みをマスクするのに対してエラールーチンはマスクしないことである。状態マシン218はいまエラーピンというよりは割り込みとして配線されている。

ジャンプ・アドレス

プログラムカウンタにロードされたアドレスはジャンプアドレスである。このアドレスの12ビットはマイクロ符号フィールドに含まれている。これは絶対アドレスであって良く、ALU222の出力によってそれを置\*

\* 換した部分を有するものであっても良い。もしアドレスを置換すべき場合は、状態マシン218は演算コア219が命令を実行し、置換のためにALU222の出力を状態マシンに供給し終えるまで待つ。

【0371】本発明によるアドレスのフォーマットが表23「ジャンプ・アドレス置換」に示されている。「a」で印をしたビットは絶対アドレスビットを示している。下位の残りのアドレスビットが置換される。「s」で印をした最下位ビットが置換ビットである。

【0372】

【表23】

置換されたビット数	B	A	9	8	7	6	5	4	3	2	1	0	S
0	a	a	a	a	a	a	a	a	a	a	a	a	0
1	a	a	a	a	a	a	a	a	a	a	a	0	1
2	a	a	a	a	a	a	a	a	a	a	0	1	1
3	a	a	a	a	a	a	a	a	a	0	1	1	1
4	a	a	a	a	a	a	a	a	0	1	1	1	1
5	a	a	a	a	a	a	a	0	1	1	1	1	1
6	a	a	a	a	a	0	1	1	1	1	1	1	1
7	a	a	a	a	0	1	1	1	1	1	1	1	1
8	a	a	a	0	1	1	1	1	1	1	1	1	1
9	a	a	0	1	1	1	1	1	1	1	1	1	1
10	a	0	1	1	1	1	1	1	1	1	1	1	1
11	a	0	1	1	1	1	1	1	1	1	1	1	1
12	0	1	1	1	1	1	1	1	1	1	1	1	1
ジャンプ・アドレス	1	1	1	1	1	1	1	1	1	1	1	1	1

表23.ジャンプアドレス置換

本発明のアドレス置換の特徴によりジャンプ・テーブルを構成することが可能になる。

状態マシン内部の命令

ステータス・バスにおいて繰り返し条件付きテストを行うことが好ましい。これらの命令は状態マシン218の内部命令であり、演算コア219からの安定したフィールド・パックを必要とする。従って、このタイプの命令は、それらの実行を失敗するであろう演算コア219に対しては無効であるとマークすることができる。よって、「有効」ビットが演算コア219に対して有効な命令をマークするために設けられている。

状態マシンテスト

本発明においては状態マシン218の動作を検証することが可能のように、多数のレジスタがマイクロプロセッサにアクセス可能である。アクセスは「アクセス」レジスタを1にセットすることにより可能になり、該レジスタがこの値をリードバックするまでレジスタをボーリングする。次に、状態マシンは停止し、安全にアクセスできる。マシンは、「アクセス」レジスタにゼロを書き込むことにより再スタートさせることができる。

【0373】マイクロプロセッサはアクセスをもつとき、それは以下のレジスタに読み出し書き込みが可能である。

—プログラムカウンタ

—コール・リターン・アドレス

—割り込みリターン・アドレス

—割り込みステータス・ビット（即ち、割り込みが進行中であるか否かを示す）

—マイクロ符号の全てのビット

表24はこれらのレジスタの種々のアドレスを示している。

【0374】状態マシン218はまた、マイクロプロセッサイベントを生成することによりそれ自身を停止することができる。イベントのマスクビットがセットされているときにのみ、マシンが停止する。その後、このイベントをサービスする時点では通常の方法でアクセスが得られる。イベントはリセットアドレス（0x00）へのコールによって生起される。このコールは実際には取り込まれないが、命令が実行された後に単にイベントを生成する。しかしながら、それは検査のために、インストラクションROMの出力に残留する。

【0375】本発明の状態マシン218はその命令を通して単一ステップで進むモードを有している。単一ステップモードではMSRレジスタのビット0をセットすることによって開始される。その後、マシンは各命令の前に停止する。停止状態は「1」=Stoppedで表さ

れる。その後、実行寸前の命令は命令ROMの出力に位置し、マイクロプロセッサのアクセスを介して変更可能である。マシンを再スタートするためには、MSSRレジスタのビット1に「1」を書き込む。これらのビットレジスタは両方が同期が取られ、従って、それらがアクセス可能となる前にマイクロプロセッサアクセスが必要である。

状態マシン・マイクロ符号マップ

表25は本発明の状態マシンのためのマイクロ符号マップを示している。

【0376】

【表25】

ビット数	2	1	0	f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0
ビット用途	a	a	a	a	a	a	a	a	a	a	a	a	s	c	condition	v			

表25 状態マシンU符号語

a = アドレス；

s = アドレス置換；

c = コール又はジャンプ；

condition = ジャンプ条件コード；

v = 演算コアに有効な命令

演算コア

本発明において、演算コア219がMSM218内における全てのデータ操作を行う。図67に示された如く、演算コア219の一般的構成は、利用可能なバスからそれらの入力を選択し、出力としてバスを提供する機能的ブロックを含む。演算コア219は32ビット幅であり、他の実装態様において8、16、24又は32ビットのデータバスを形成することが可能なビットスライスから構成される。

【0378】図68に図示された如く、本発明の演算コア219は3つの主要な機能的ブロック、即ちデータストリームと通信するためのトークンポート360、演算（或いは他の機能）を実行するためのALU222、全てのレジスタを含むレジスタファイル221を有する。図68においては全ての出力バスに符号が付けられている。ブロックへの入力はこれらのバスから選択される。これらのセレクタのサイズ及びそれらの入力の変更可能であり、マイクロ符号により制御される。

ALU

本発明によるALUブロック222は、演算コアにおける全ての演算及び数の操作を担当する。それは非常に複雑な演算（循環、乗算、及び除算等）を比較的単純な動作（即ち、シフト、条件付反転及び加算）等の組み合わせにより実行することを可能にする。これらのブロックの各々について以下に記述する。その後、これらが演算コア219において、全体として、いかにしてより複雑な演算を実行するためにも利用されるかについて例が示されている。

シフトブロック

アドレス	用途
0x000	リセットアドレス
0x001	割り込み/エラーアドレス
0x002	U符号プログラムアドレス
-0xfff	

表25 状態マシンU符号マップ

状態マシン・マイクロ符号語

同様に、表26は本発明による状態マシン・マイクロ符号語を示している。

【0377】

【表26】

ビット数	2	1	0	f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0
ビット用途	a	a	a	a	a	a	a	a	a	a	a	a	s	c	condition	v			

表26 状態マシンU符号語

本発明において、「シフト」ブロックは1つのビットが左シフト、右シフト、またはシフトしないことを可能にする。1ビットバスKは、あたかも付加ビットであるかのようにワード内で回転する。これは表27に示されている。

【0379】

【表27】

ss	シフト機能
00	l' = l
01	l' = l; NOP
10	l' = (l << 1) + K
11	l' = (l >> 1) + (K << 2)

表27 シフトブロック

もしss=0b01aであれば、「NOP」がALU222全体に送信される。これは動作無しであり、最後の動作から、どの状態フラグが変更され始めることも防止する。

キャリーブロック

キャリーブロックは状態レジスタからキャリービットを取り込むか又はそれをクリアする。単一ワードの付加及びその後の動作において、キャリービットはクリアされるけれども、複数ワードの動作においては、前の動作により生成（そして状態フラグに格納された）されたキャリーがキャリーとして使用される。これは表28に示されている。

【0380】

【表28】

c	キャリー機能
0	OC=0
1	C=状態フラグからのH

表28 キャリーブロック

条件ブロック

50 本発明によるブロック条件、被加数、ALUコア機能へ

のキャリーが表 29 に定義されている。

【0381】

【表 29】

ii 逆の機能	
00	J' = J C' = C
01	J' = -J C' = -C
10	J' = J & L C' = C & L
11	J' = (L?J : -J) C' = (L?C : -C)

表 29 条件ブロック

ALU コア

本発明の ALU コア 222 は 2 の補数の演算を用いて単純な論理セット及び演算機能を行う。これらは表 30 に定義されている。

【0382】

【表 30】

ff	ALU コア機能
0	I' * J' * C' Add
1	I' * J' XOR
10	I' & J' AND
11	I'   J' OR

表 30. ALU コア

ALU コア 222 の結果から 4 つの状態フラグが生成される (表 31 参照)。

【0383】これらはレジスタファイル 221 に格納され (表 36 に示された如く)、条件コードと比較するために状態マシン 218 に送り返される。

【0384】

【表 31】

意味	逆の機能
Carry	ALU 動作からのキャリー出力
Zero	ALU の結果がゼロ
Negative	ALU の結果の MSB = 1
Overflow	ALU 動作がオーバーフロー

表 31 ALU コアにより生成される状態フラグ

ALU マイクロ符号語

表 32 が ALU マイクロ符号語を示している。

【0385】

【表 32】

ビット 数	6	5	4	3	2	1	0
ビット用途	s	s	l	l	f	f	c

表 32 ALU マイクロコード語

上において、

s s はシフトブロック制御

i i は条件ブロック制御

f f は ALU コア制御

c はキャリーブロック制御である。

ALU の使用

表 33 は本発明による、ALU の種々の機能のためのビットパターンを示している。

【0386】

【表 33】

ビット数	6	5	4	3	2	1	0
加算 (I+J)	0	0	0	0	0	0	0
減算 (I-J)	0	0	0	1	0	0	0
乗算	1	0	1	0	0	0	0
除算	1	0	1	1	0	0	0

レジスタファイル

図 69 は本発明のレジスタファイル 221 を図示している。レジスタファイル 221 は 64 個の 32 ビット語レジスタを含んでいる。レジスタファイル 221 は部分ワードをアドレス指定すること、即ち、ファイルは 64 × 32 ビット、128 × 16 ビット、250 × 8 ビット、512 × 4 ビット、1024 × 2 ビット、又は 2048 × 1 ビットフォーマットとしてアドレス指定することができる。アドレスはマイクロ符号から直接的に与えられるか、又はアドレスはその一部分を空間レジスタから置換したものであっても良い。これによりレジスタの指標付きアクセスが可能になる。

【0387】各命令において、読み出し・修正・書き込みが単一のレジスタ上で行われる。読み出し・修正・書き込みにより部分ワードをファイルに書き戻すことが容易になる。書き込みのソースはそれ自身の独立したマイクロ符号を有する外部のマルチプレクサによって決定される。もし書き込みが要求されない場合には、レジスタファイル 221 の出力がマルチプレクサによって選択されなければならない。

【0388】部分語 (部分ワード) はモードレジスタのビット 0 に応じて符号付き又は符号無しの数として扱われる。もし部分語が負であれば (即ち、その MSB がセットされていれば)、それはバスの全幅まで符号が拡張される。これにより演算において部分語を容易に使用することが可能になる。また、本発明のレジスタファイル 221 内の 3 つのローケーションが専用バスに接続されているけれども、尚それらは、他のレジスタファイルのローケーションと共に使用することができるように成されている。これらは A 及び B レジスタ、及び図 69 に示されたステータスレジスタである。レジスタファイルはまた、付随するターミナルカウント・レジスタ、定数レジスタ及びレジスタファイルのモードを指定するモード・レジスタとのアドレス置換のためのインデックス・レジスタを含んでいる。

レジスタファイル・アドレス指定

本発明によれば、アドレス指定は二つの異なる特徴に対してする必要がある。語の変化する幅の部分にアクセスす

るための可変長アドレスとアドレス置換である。部分語をアドレス指定するためには長いアドレスが必要である。従って、全てのアドレスは可変長であり、以下の如く符号化される：「a」がアドレスビットであるとき、\*

\* アドレスビットの最下位は「S」、置換ビットである。

【0389】

【表34】

データ幅	B	A	9	8	7	6	5	4	3	2	1	0	S
1	1	a	a	a	a	a	a	a	a	a	a	a	a
2	0	1	a	a	a	a	a	a	a	a	a	a	a
4	0	0	1	a	a	a	a	a	a	a	a	a	a
8	0	0	0	1	a	a	a	a	a	a	a	a	a
16	0	0	0	0	1	a	a	a	a	a	a	a	a
32 (24)	0	0	0	0	0	1	a	a	a	a	a	a	a

表34 可変長アドレス指定

アドレス指定は大きなエンディアンである。換言すれば、高位である程、語のより高位の部分が高いアドレスによってアドレスされる。

【0390】アドレスの部分「a...a」はインデックスレジスタの一つによって置換することができる。表34に一例として定義された8つのビット語の一つのア※

※ ドレスを用いて、表35は置換すべき最下位ビットの数をどのようにして定義するかを示している。全ての後端のゼロは置換される。

【0391】

【表35】

置換すべきビット	C	B	A	9	8	7	6	5	4	3	2	1	0	S
0	0	0	0	1	a	a	a	a	a	a	a	a	a	0
1	0	0	0	1	a	a	a	a	a	a	a	a	0	1
2	0	0	0	1	a	a	a	a	a	a	a	0	1	1
3	0	0	0	1	a	a	a	a	a	a	0	1	1	1
4	0	0	0	1	a	a	a	a	a	0	1	1	1	1
5	0	0	0	1	a	a	a	a	0	1	1	1	1	1
6	0	0	0	1	a	a	a	0	1	1	1	1	1	1
7	0	0	0	1	a	a	0	1	1	1	1	1	1	1
8	0	0	0	1	a	0	1	1	1	1	1	1	1	1

表35 アドレス置換

例えば、4ビットを32ビットアドレス内に置換すると、0b000001aaaa011111の形になり、ゼロビットを1ビットアドレスに置換すると0b1aaaaaaaaaaa0になる。

【0392】本発明において、置換は二つの8つのビットインデックスレジスタのレジスタファイルマイクロ符号語において指定される一つから来ている。従って、最大8つのビットが一つのアドレス内に置換可能であることが分かる。また、上記の方式によって、0b00000000000000又は0b1111111111111111等の不法なアドレスを使用することが可能であることが分かる。不法なアドレスはアドレスがアクセスされない結果となり、レジスタファイルの出力バスの状態が不明にする。

レジスタファイル・レジスタタイプ

本発明において、複数のレジスタファイル・レジスタタイプが存在する。その各々が以下に記述されている。

【0393】・ 独立にバスを有するレジスタ

3つのレジスタ (A、B、及びステータス・レジスタ) はそれらの専用のバスを有し、またレジスタファイルに

30 おいて通常の方法でアクセスすることができる。これによりレジスタは演算コア219においてより多くの位置で接続され得ると共にレジスタファイル内の他のレジスタと平行してアクセスすることが可能になる。独立バスはアクセスがその全幅、即ち32ビット幅でのみレジスタにアクセス可能である。

【0394】これらのレジスタにはマイクロ符号ライトイネーブルは存在しない。これらへの書き込みはそれ自身のマイクロ符号制御語を有する外部マルチプレクサを介してのみ可能である。書き込みを阻止するためにそれらは、図70に示された如くそれ自身の値を用いて書き込みされることが必要である。独立バスレジスタがレジスタファイル内の如く書き込みされるときには、独立バス書き込みは抑圧される。

【0395】ステータスレジスタは独立バスレジスタとして実動化される。レジスタのビットは表36に定義されている。

【0396】

【表36】



133		134	
ビット	意味	注 釈	
0	1 (イデックス・レジスタ)	そのターミナル・カウントを通過したインデックスレジスタインクリメント	
1	E 拡張	入力からの拡張ビット	
2	V オーバーフロー	ALUの動作がオーバーフロー	
3	N 負	ALUの結果のMSB=1	
4	Z ゼロ	ALUの結果がゼロ	
5	C キャリー	キャリー from ALU 動作	
6	Gnd	未使用	
7	Gnd	未使用	

表36 ステータス・レジスタの定義

・ インデックス及びターミナル・カウント・レジスタ二つの8つのビットインデックスレジスタがアドレスへの置換のために設けられているこれらの内の一つはマイクロ符号の制御に従って、命令毎にインクリメントされる。更に、各々にはターミナル・カウント・レジスタが付随している。レジスタインクリメントが渡されたときには、そのターミナル・カウントがゼロにリセットされる。

【0397】インデックス・レジスタはY及びZと呼ばれ、ターミナル・カウント・レジスタU及びVをそれぞれ有する。これらの全てはレジスタファイル内でアクセスすることができる。インデックスレジスタZはその出力に接続された予め定義されたデコーダ（今のところ、この復号は反転である）を有する。モードレジスタ（ビット1）におけるIndex\_Modeに応じて、インデックスレジスタよりはむしろこのデコーダが、アドレス置換及びレジスタファイルにおけるレジスタZからの読み出しに用いられる。（Index\_Mode=1はリード・デコード、Index\_Mode=0はリード・カウント）

・ カウント）

\* 定数レジスタ

\* 本発明において、レジスタファイルの32ビットロケーションの内の16は予め定義された定数である。これらは通常のレジスタとして読み出すことが可能である。これらのロケーションへの書き込みは何も効果を生じない（現実実施例の為に選択された定数は0-7であるけれども、他の数値の定数を用いても良いことは明かである。）

本発明によるこの定数の実装は、マイクロ符号における一定のフィールド及び演算コアにおける一定のパスの必要を無くするものである。しかしながら、これはプログラムにおいて使用可能な定数の数を制限する。（数16は得ることができる。）これらの定数はその時点毎にプログラムされる。更に、もし必要であれば、しばしば用いる値はマルチプレクサに結合しておくことが可能である。

レジスタファイル・アドレスマップ

表37は本発明のためのレジスタファイルアドレスマップを示している。

【0398】

【表37】

32ビットロケーション	ビット	レジスタ
0x00	全	Aレジスタ
0x01	全	Bレジスタ
0x02	7:0	ステータス・レジスタ
0x02	8	符号拡張モード
0x02	9	インデックス復号モード
0x02	31:10	通常レジスタ
0x03	7:0	Yインデックス・レジスタ
0x03	15:8	Zインデックス・レジスタ
0x03	31:16	通常レジスタ
0x04	7:0	Uターミナル・カウント・レジスタ
0x04	15:8	Vターミナル・カウント・レジスタ
0x04	31:16	通常レジスタ
0x05-0x37	全	通常レジスタ
0x37-0x3F	全	定数

表37 レジスタファイル・アドレスマップ

レジスタファイル・マイクロ符号語

表38は本発明のためのレジスタファイル・マイクロ符号

語を示している。

【0399】

【表38】

ビット番号	d	c	b	a	9	8	7	6	5	4	3	2	1	0
ビット用途	a	a	a	a	a	a	a	a	a	a	a	s	r	l

表38 レジスタファイルU符号語

a = レジスタファイルアドレス全体（常に12ビット）

s = 置換ビット

r = 置換のために用いるインデックス・レジスタ；もし n がそれぞれ 0、1 であれば Y、Z インデックス・レジスタを選択する。

【0400】l = r により指定されるインクリメント・インデックス・レジスタ

トークンポート 本発明のトークンポートは演算コアのデータストリームへの接続であり、2線式インターフェースによる接続である。トークンポート入力におけるデータはトークンポート読み込みサイクルの間にのみ定義される。従って、それは読み込みサイクルの間にのみ使用しなければならない。

【0401】もし入力ポートが読み込みサイクルの間に有効なデータを含まない場合、または書き込みサイクルの間に出力ポートが受け入れない場合には、演算コアは停止する。よって、演算コアは何も動作を行わず、マイクロ符号語を新しく読み込まず、どのレジスタにも書き込まないことになる。演算コアはこれらの条件が存在しないときにのみ再スタートする。

トークンポート・マイクロ符号語

表39はトークンポート・マイクロ符号語を図示している。

\* 【0402】

【表39】

ビット番号	1	0
ビット用途	1	0

表39 トークンポートU符号語

I = 入力ポートへの読み込み

O = 出力ポートからの書き出し

マルチプレクサ

ブロックのソースの選択はマルチプレクサを用いて実行される。殆どすべてのパスの組み合わせ（機能的ブロック例えば、ALUへの入力は記憶ブロック、例えばトークンポート又はレジスタファイルからのものであることの必要を例外として）が許される。

20 【0403】マルチプレクサは2、4又は8入力の何れかである。それらは、従って、マイクロ符号語の1、2又は3ビットをその入力の選択を制御することのためにそれぞれ使用する。

MPIメモリーマップ

表40は本発明によるMSMアドレスマップを示している。

【0404】

【表40】

アドレス	ビット	ロケーション
0x000	0	MSM偶数ビット
0x001	0	MSMマスクビット
0x100	7	アクセスビット
0x101	0	MSSRセットシングルステップ
0x101	1	MSSRモニタ・シングル・ステップ
0x101	2	MSSR割り込みステータスレジスタ（リード・オンリー）
0x102	3:0	プログラムカウンタMSB
0x103	7:0	プログラムカウンタLSB
0x104	3:0	リターン・アドレスMSBコール
0x105	7:0	リターン・アドレスLSBコール
0x106	3:0	割り込みリターンアドレス
0x107	7:0	割り込みリターンアドレス
0x200-0x2ff	7:0	レジスタファイル

表40 MSMアドレスマップ

序文

MPEG符号化の標準（MPEG-1及びMPEG-2共通）においては量子化された係数は「イベント」としてコード化される。各イベントはRUN及びLEVELとしてコード化される。RUNは与えられたゼロ以外の係数に先行するゼロ係数の数であり、LEVELはその

係数の値である。加えて、一つの特別のイベント、End-of-Block、がブロックの残りは全てゼロであることを示すために、最後のゼロ以外の係数の後に使用される。

【0405】例えば、以下の係数のシーケンスを仮定すると、

137

1、-7、0、3、0、0、-1、0、0、0、  
0、...、0（合計64の係数）  
これらは（RUN、LEVEL）として表される以下の  
イベントとしてモデル化される。  
【0406】（0、1）（0、-7）（1、3）（3、  
-1）（EOB）  
モデル化処理の逆を実行して64の係数の各々が以降の  
処理のための単純な数として表されるようにすることは  
逆モデラーの仕事である。

インターフェース

以下の信号ピンが本発明の逆モデラーにデータを転送す  
るために使用される。:

・LEVEL [11:0]  
・RUN [5:0]  
・in\_extn  
・in\_valid  
・in\_accept

トークンはlevel [11:0] バス（下位の8ビッ  
ト: level [7:0]）上で転送される。

【0407】run [5:0] RUN情報を担う補助の  
バスとして動作する。それはDATAトークンのデータ  
語における場合を除いて特に意味を持たない。以下の信  
号が逆モデラーの出力において使用される。

・out\_data [11:0]  
・out\_extn  
・out\_valid  
・out\_accept

機能の記述

逆モデラーの出力に現れるデータトークンにおいて常に6  
4の係数が存在するようにDATAトークン内のデータ  
は拡張されている。多くの場合、DATAトークンの最  
後のデータ語は64番目の係数を生成しない。これはエ  
ラーではなく、この時点でEOBまでビットストリ  
ーム内にコード化されてしまったためである。従って、  
この状態においては逆モデラー合計64の係数が出力に  
おいて生成されるまでゼロデータトークン語を出力し続け  
なければならない。

【0408】特定の状況（例えば、データエラーが生じ  
た時など）においては、逆モデラーへの入力におけるデ  
ータトークンが64以上の係数を表すことが有る。この  
状態において、モデラーは全ての余分のデータを廃棄し  
、ちょうど64の係数を含むトークンをその出力に生  
成しなければならない。入力に現れるすべての非データ  
トークンは、変更無しに逆モデラーの出力に単に転送さ  
れる。

【0409】タイミング要件

本発明においてはデータがクロックレートで逆モデラー  
を通過することが要求される。Imodelへの入力に  
おいてギャップが存在せず、出力に接続された回路がI  
modelを停止させることがない（即ち、in\_v a 50

138

lid=1、out\_accept=1）状態において  
は、新しいデータ語がクロックサイクル毎にImode  
lの出力に出頭する。しかしながら、この状態において  
は、ゼロ以外のRUN（データトークンにおける）は各  
入力に対して1以上のデータ語が生成させるため、Im  
odelは1クロックサイクル毎に新しいデータをその  
入力に受け付けなければならないことに注意すべきであ  
る。

【0410】マイクロプロセッサ・インターフェースア  
クセス

本発明の逆モデラー回路はその通常の動作のモードにお  
いてMPIに接続されている必要はない。エラー条件  
（係数過多）がマイクロプロセッサ割り込みを生成すべ  
きでないことに注意すべきである。それは単に余分のデ  
ータを廃棄することにより内部的に処理される。

【0411】しかしながら、ブロックの入力におけるス  
ノープ（テスト）回路のためにはマイクロプロセッサア  
クセスが必要とされる。

序文

MPEG符号化の標準において、係数は「ジグザグ」ス  
キャンされ、低周波数の係数は高周波数の係数の前に伝  
送される。

【0412】本発明による逆ジグザグの機能は、逆モ  
デラーからそれが受け取る係数の一次元ストリームをID  
CTによって処理することが可能係数の二次元アレ  
イに変換することである。MPEG-1においては、ただ  
一つのスキャンパスが使用されており、これは文字ど  
おりジグザグ（従って、その名前が）であった。しかしな  
がら、MPEG-2は二つのスキャンパスを使用する。  
第1のスキャンパスはもともとMPEG-1パスであ  
り、第2のスキャンパスは、著しく大きな垂直周波数構  
成部品が存在しがちであるインターレース符号化におい  
て使用するために最適化されている。

【0413】明らかにジグザグスキャンの順序で伝送さ  
れる係数に加えて、量子化マトリクスも同様にジグザグ（I  
DCTの一部として実現されている）の前に位置してい  
る。従って、量子化器はダウンロードされた量子化マ  
トリクス係数と同一の順序で到着する係数の一次元スト  
リームに対して動作する。よって、量子化器は単に第1の  
係数を第1のマトリクスエレメントに関係づけ、第2の  
係数を第2のマトリクスエレメントに関係づけ、以下同  
様の動作を行うことが必要である。

【0414】しかしながら、MPEG-2においては  
二つのスキャンパスが有るため、本発明においては新  
しいアプローチが行われており、逆ジグザグが逆量子化  
器に先行する構成である。係数及びダウンロードされた  
マトリクスは両方とも逆スキャンされ、逆量子化器はこ

ここでは二次元データに対して動作する。このことは、データの全ての3つの表現（IZZの出力における二つのジグザグスキャンおよびラスタースキャン順序）において、最初の係数は常に最初となり、最後の係数は常に最後となることによってのみ可能であることに注意すべきである。第1の係数はDC項であるため、逆量子化器Iquantにおいて特別に扱われる。最後の係数は全ての他の係数の値の関数としてのミスマッチ制御の結果によって修正することが必要になる可能性があるため（故にそれは最後でなければならない）、特別に扱われる。残りの62の係数は全て（各々がそれ自身の量子化マトリクスエレメントを有することを除いて）同一の方法で扱われる。

【0415】インターフェース  
以下の信号が本発明の逆ジグザグの入力において使用される。

- ・in\_data [11:0]
- ・in\_extn
- ・in\_valid
- ・in\_accept

以下の信号が逆ジグザグの出力において使用される。

- 【0416】
- ・out\_data [11:0]
- ・out\_extn
- ・in\_valid
- ・out\_accept

機能の記述

逆ジグザグIZZは以下のトークンに応答する。

- 【0417】
- ・PICTURE\_START
- ・ALTERNATE\_SCAN
- ・DATA
- ・QUANT\_TABLE

他の全てのトークンはIZZ内を修正されることなく通過する。

【0418】PICTURE\_STARTトークンはIZZに対して二つのスキャンパスのどちらが実行されているかを表すその内部のステート（例えば、alternate\_scan）をゼロ（MPEG-1スキャンを表す）にリセットさせる。ALTERNATE\_SCANはマスク0x6を伴い0x6が割り当てられることが可能なトークンである。ALTERNATE\_SCANトークンは表41に示されている。

【0419】

表41】

E	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	1	s

表41 Alternate\_Scan トークン

「s」はどのスキャンを以降のデータトークンのために

使用すべきを示しており、従って、IZZレジスタ"alternate\_scan"にロードされる。

【0420】DATA（データ）トークンはalternate\_scanの設定に拘らずスキャンパスゼロ（MPEG-1スキャンパス）に従って並べ替えられる。alternate\_scanはその値がどのような値であったとしてもそれを保持していること（即ち、ゼロにセットしてはならない）が必要であり、それによって以降のデータトークンは正確に扱われることに注意すべきである。

【0421】QUANT\_TABLEトークンはalternate\_scanの設定に拘らずスキャンパスゼロ（MPEG-1スキャンパス）に従って並べ替えられる。alternate\_scanはその値がどのような値であったとしてもそれを保持していること（即ち、ゼロにセットしてはならない）が必要であり、それによって以降のデータトークンは正確に扱われることに注意すべきである。誤りを含んで形成されたトークンDATA及びQUANT\_TABLEトークンはどちらも誤りを含んで形成される可能性が有る。データトークンは、逆モデラーImodelがそれが正確に形成されていることを確認しているため、正確なものであることは明らかである。しかしながらQUANT\_TABLEについてはそのような保証は無い。誤りを含んで形成されたQUANT\_TABLEトークンの処理を実現しなければならないため、同様にデータトークンについてもその処理を実現しなければならない。本発明によれば、DATA及びQUANT\_TABLEトークンがIZZへの入力において出現するときに過度に短くときにも、出力においてトークンは正確な数（64）のデータ語となる結果でなければならない。これらの語に含まれるデータは重要ではなく、おそらくトークンの開始以前にリオーダーリングRAMににたまたま存在していた無用な値となる。同様に、長過ぎるDATA及びQUANT\_TABLEトークンも出力において正確に形成されたトークンとなる必要がある。最初の64係数（マトリクスエレメント）が使用されねばならず、残りは廃棄されねばならない。

【0422】誤りを含んで形成されたトークンに続く、以降の全ての（正確に形成された）トークンは正し扱われねばならない。マイクロプロセッサインターフェースエラー（割り込み）が生成されることは要求されない。

ラスタースキャン順序

IZZの出力において、本発明によるDATA及びQUANT\_TABLEトークンは二次元データを表している。しかしながら係数は実際にはまだ数値の一次元系列として転送されている。ここで、データを行として転送すべきかあるいは列として転送すべきかNの問題が生じる。

【0423】予測回路はベル領域のデータがラスタースキャン順序で構成されることを要求する。IDCTがデータを転置するため、IDCTに入るデータは逆の順序でなければならない。表42にはDATA及びQUANTIZATION TABLE トークンのためにIZZの出力に転送される係数の順序を示されている。

【0424】

【表42】

増加水平周波数 -U										
0	0	1	2	3	4	5	6	7		
0	0	8	16	24	32	40	48	56		
1	1	9	17	25	33	41	49	57		
2	2	10	18	26	34	42	50	58		
3	3	11	19	27	35	43	51	59		
4	4	12	20	28	36	44	52	60		
5	5	13	21	29	37	45	53	61		
6	6	14	22	30	38	46	54	62		
7	7	15	23	31	39	47	55	63		

表42 IZZ出力係数

マイクロプロセッサ・インターフェース・アクセス IZZの通常の機能においてはマイクロプロセッサ・アクセスは要求されない。しかしながら、おそらく、リオーダーリングRAMをテスト可能にするためにはアクセスが必要とされる。また、スノープが必要とされないことも予想される。Modelの開始において一度行うことで両方のブロックのために充分である。

【0425】序文

この節においては、予測について扱う。可能な全ての予測モードは番号付けされており、その各々には何が行われなければならないかについて正確に説明するために図面が設けられている。この節の全体にわたって、半ペル・フィルタリング等の水平次元に生起する動作について特別な注意は払われていない。これは、これらの動作はRollにおける対応する動作と同一であるからである。しかしながら、垂直次元においては、インターレースされた画面フォーマットであるために事情は大きく異なっている。

フレーム画面における予測

フレームに基づく予測

このモードにおいては、基準フレームから予測が生成される。その結果は、二つの基準フィールドが始めに一つのフレームに合成され、そのフレームから予測が行われるかの如くである。これは正確に、Rollにおいて記述された状態であることに注意すべきである。

【0426】半ペル・フィルタリングは垂直方向において行うことができ、これはベクトルの最下位ビットによってトリガーされる。最下位ビットに加えて、次の最上位ビット（ビット1）は、予測の最上ラインが上端基準フィールドに由来するの、あるいは下端基準フィールドに由来するのかを決定するものであるから、特に重要である。

【0427】これによって、4つの場合を考慮せねばならず、各々が垂直ベクトルの最下位の二つのビットのバイナリ値に依存する。

ベクトル [1] = 0、ベクトル [1] = 0

図71に示された如く、半ペル・フィルタリングが無い場合、ちょうど16ライン（クロマについては8）が読まれ、各基準フィールドから8（4）ラインが読まれる。

【0428】ベクトル [1] = 0、ベクトル [0] = 1 同様に、図72に示された如く、17（9）ラインが読まれ、9（5）ラインが上部（トップ）基準フィールドから読まれ、8（4）ラインが下部（ボトム）基準フィールドから読まれる。

ベクトル [1] = 1、ベクトル [0] = 0

再び、図73に示された如く、ちょうど16（8）ラインが読まれるけれども、ここでは予測の最上ラインが下部基準フィールドから読まれることに注意すべきである。

【0429】ベクトル [1] = 1、ベクトル [0] = 1 そして、図74は17（9）ラインが読まれ、8（4）ラインが上部基準フィールドから読まれ、9（5）ラインが下部基準フィールドから読まれることを示している。よって、ビット1はどちらの基準フィールドが予測を生成するために読むことが必要な最上ラインを保持しているかを示している。加えて、もしビット0がセットされているか、それはどちらの基準フィールドが半ペル・フィルタリングの実行を可能にする余分のラインを有しているかを示している。

【0430】両方のフィールドがDRAMから読み出されるまで半ペル予測は実行できないことが明かである。フィールド・ストアにおけるオフセットを得るために垂直動きベクトルをスケールリングするときには大きな注意を払わねばならない。以下の表、表43、がその効果を表している。

【0431】

【表43】

10

20

30

40

ベクトル	ビットパターン	フィールドにおけるオフセット	
		上部フィールド	下部フィールド
-2	...11100	...11110 (-2)	...1111 (-2)
-1.5	...11101	...11111 (-1)	...11110 (-2)
-1	...11110	...11111 (-1)	...11111 (-1)
-0.5	...11111	...00000 (0)	...11111 (-1)
0	...00000	...00000 (0)	...00000 (0)
0.5	...00001	...00001 (1)	...00000 (0)
1	...00010	...00001 (1)	...00001 (1)
1.5	...00011	...00010 (2)	...00001 (1)
2	...00100	...00010 (2)	...00010 (1)

フィールドに基づく予測（フレーム画面内）

このモードにおいて、各フィールドは独立して扱われる。別々のベクトルが二つのフィールドの各々に用いられる。各ベクトルには予測が上部基準フィールドから行われるべきか或いは下部基準フィールドから行われるべきを示す追加の単一ビットフラグ（motion\_vertical\_field\_select）が関係づけられる。ベクトルの最下ビットはなお半ペル・フィルタリングが必要であることを示しているけれども、ビット1に何にも特別な意味は無い。フィールド・ベクトルはフレーム・ベクトルとは別の単位で測るものであり、フィールド・ベクトルの値がnである場合には、値が2nのフレームベクトルと同一の実際の変位（ガラス）を表すことになることに注意すべきである。

【0432】この場合、しかしながら、（4つの境界変数、即ち二つのベクトルの各々のためのmotion\_vertical\_field\_select及び二つのベクトルの各々のためのビット0、が存在するため）16の場合について考慮しなければならない。描画すべき場合が非常に多く存在するため、以下の図では上部フィールドの予測についてのみ扱う。下部フィールドは同様の方法によって得ることができる。

【0433】図75に図示された如く、motion\_vertical\_field\_select=0、ベクトル[0]=0の場合、8（4）ラインが上部基準フィールドから読み出されて予測の上部フィールドが形成される。図76はmotion\_vertical\_field\_select=0、ベクトル[0]=1の場合を示し、9（5）ラインが上部基準フィールドから読み出されて、次にそれらは半ペル・フィルタリングされて予測の上部フィールドを形成する。

【0434】同様に、図77はmotion\_vertical\_field\_select=1、ベクトル[0]=0の場合を示している。8（4）ラインが下部基準フィールドから読み出されて予測の上部フィールドが形成される。そして、図78はmotion\_vertical\_field\_select=1、ベクトル[0]=1の場合を示している。

【0435】9（5）ラインが下部基準フィールドから

読み出されて、次にそれらは半ペル・フィルタリングされて予測の上部フィールドが形成される。

デュアル・プライム（フレーム画面における）

デュアル・プライムは前節のフィールドに基づく予測の特別な場合である。デュアル・プライムは本質的に、二つの特徴を組み合わせたものである。

【0436】・4つの独立した（それらは各々が異なるベクトルを有するという意味で独立した）フィールド予測が形成されるという事実にも拘らず、事実上た一つの動きベクトルが伝送されるための特別なベクトル符号化方法。こうして、ベクトル・オーバーヘッドが劇的に減少する。

・各フィールドに対して、予測情報が基準フィールドの各々から読み出される。次にこれが平均化されて最終的な予測が形成される。このことは、別々の前方及び後方予測が行われて次に平均化されるB-1画面の場合に非常に類似している。本発明において、ベクトルの符号は全てP-1画面内で実行される。よって、予測回路がデータを受け取るとき、実際に4つの別々のベクトルが存在する。

【0437】デュアル・プライム平均化はB-1フレーム平均化回路を再使用することにより実行される（デュアル・プライムそれ自身はB-1フレームにおいて用いることはできない）。予測回路に関する唯一の複雑性は、後方予測（後方ベクトル・トークン等を使用した）は前方基準フィールド（後方基準フィールドに対する）から実行されねばならないことを示す信号方式に關係している。P-1画面は通常、後方予測を要求する事は全く無い。P-1画面は通常、後方予測のためにどちらの基準ストアを使用するかを決めることができるように、画面タイプ（P又はB）の記録を保持することのみが必要である。

画面フィールド内の予測

フィールドに基づく予測

これは、フレーム画面内のフィールドに基づく予測に非常に類似しており、motion\_vertical\_field\_select及び動きベクトルの最下ビットに依存して4つの場合が存在する。予測は単にデコードされている画面（全上部フィールド又は全下部フィ

ールドの何れかである) のためであるから、形成された予測内の上部フィールド及び下部フィールドを議論することは実際に適当でないことに注意すべきである。

【0438】図79はmotion\_vertical\_field\_select=0、ベクトル[0]=0の場合を示している。16(8)ラインが上部基準フィールドから読み出されて予測が形成される。図80はmotion\_vertical\_field\_select=0、ベクトル[0]=1の場合を示している。

【0439】17(9)ラインが上部基準フィールドから読み出され、半ペル・フィルタリングされて予測を得る。図81はmotion\_vertical\_field\_select=1、ベクトル[0]=0の場合を示している。16(8)ラインが下部基準フィールドから読み出されて予測が形成される。

【0440】図82はmotion\_vertical\_field\_select=0、ベクトル[0]=1の場合を示している。17(9)ラインが下部基準フィールドから読み出され、半ペル・フィルタリングされて予測を得る。

16×8MC

このモードにおいて、全体のブロックは、一方が他方の上にある二つの16×8領域に分割されている。各領域に対して、別々のフィールド・ベクトルが伝達される。ここでも再び、16の場合を考慮しなければならない(4つの2進変数、即ち二つのベクトルの各々に対するmotion\_vertical\_field\_selectと、二つのベクトルの各々に対するビット0が存在するため)。また再び、図示すべき場合が多いため、以下の図では上位の16×8領域についてのみ扱う。下位の領域は同様の方法で得ることができる。

【0441】図83はmotion\_vertical\_field\_select=0、ベクトル[0]=0の場合を示している。8(4)ラインが上部基準フィールドから読み出されて上位16×8領域の予測が形成される。図84はmotion\_vertical\_field\_select=0、ベクトル[0]=1の場合を示している。

【0442】9(5)ラインが上部基準フィールドから読み出されて半ペル・フィルタリングされ、上位16×8領域の予測が形成される。図85はmotion\_vertical\_field\_select=1、ベクトル[0]=0の場合を示している。8(4)ラインが下部基準フィールドから読み出されて上位16×8の予測が形成される。

【0443】図86はmotion\_vertical\_field\_select=1、ベクトル[0]=1の場合を示している。9(5)ラインが下部基準フィールドから読み出されて半ペル・フィルタリングされ上位16×8の予測が形成される。

フィールド画面におけるデュアルプライム

フィールド画面におけるデュアルプライムは単にフィールド画面におけるフィールド予測の特別な場合である。2つのフィールドベクトルが使用される(一方が上部基準フィールドを参照し、他方が下部基準フィールドを参照し、パサがこれを確認する)。予測の一方が後方予測を生成するように思われるけれども、これはP-画面であるから、予測回路はこれを第2の前方予測として解釈する。その結果の二つの予測はB-フレーム平均化において用いられたものと同じの回路を用いて平均化される。

【0444】全体的組織

図87は、本発明によるディスプレイパイプラインの全体的組織を示している。データはDRAMインターフェースから単一の多重化インターフェースに到来する。更に、DRAMインターフェースはデータを正確なバイト数の上の次の32バイトの境界まで切り上げられたラインとして供給する。しかしながら、ラインの終端に向けて位置するペルは意図したディスプレイ領域の外側になる可能性がある。

【0445】データに加えて、DRAMインターフェースは各チャンネル(Y、Cr及びCb)で、そのバイトが現在のディスプレイラインの最後であるか否かを表すビットを供給する。更に、データがどのフィールドから来たものであるかを表すビットが供給される。本発明のディスプレイパイプライン内の第1のブロックは3つのチャンネルに分かれる。色(Cr及びCb)データは垂直アップ・サンプラー210に供給される。輝度(Y)データは、もし必要であれば、FIFO内で遅延される。

【0446】垂直アップ・サンプラー210は色データを2:1の割合でアップ・サンプリングして輝度データと同じ数の色データのラインがあるようにする役割を担っている。これを行うために垂直アップ・サンプラーは色データの各ラインを格納し、このライン及び以降のラインの間に間隔を設けた出力ペルを生成する。次のディスプレイ・パイプライン内のステージは"水平整列器370"と表示されている。これは、水平アップ・サンプラー212の部分として実現されており、その役割は、データを整列させて各ラインの始端において、3つのチャンネルの各々の第1のペルが水平アップ・サンプラー212に正確に供給されるようにすることである。各ラインの終わりにおいては、一般に、各チャンネルは異なる時間に"データが無くなる"ことが予想される。"水平整列器"ブロック370は余分のデータを有するチャンネルからのこの余分のデータを廃棄するとともに他のチャンネルを機能停止させ、3つのチャンネルの全てが位置合わせされて次のディスプレイラインの開始が可能な状態にすることが仕事である。

【0447】本発明において、水平アップ・サンプラー212はデータを水平にアップ・サンプリングしてデー

タを引き延ばし、テレビジョン画面のガラスを埋める。シリコンの面積を節約するために、3つのチャンネルでフィルタは共有されている。フィルタの合計の出力速度は27Mバイト/s(クロックレート)でなければならぬため、これを行うことが可能である。データはC C I R 6 0 1の順序で多重化され、それにより生成されたデータストリームは単純に最終的なデータストリームに多重化される。

【0448】水平アップ・サンプラー212は単にDRAMインターフェースにより供給される量のデータを取り込み、それを選択された率でスケーリング(基準化)することに注意すべきである。一般的に、ラスタスにおける実際のライン長に対してそれらが生成するデータは少なすぎるか又は多すぎるかである。これは出力マルチプレックスにおいて処理される。また、“水平整列器”ブロック370は完全なラインに対して各チャンネルの幾つのペルが要求されるかについては知る必要が無いことに注意すべきである。アップ・サンプリングフィルタに対する入力ペル数と出力ペル数の関係は非常に単純では無いから、この数を計算することは容易でない。水平整列器ブロック370は“要求に応じて”単にデータをその3つのチャンネルの各々の水平アップ・サンプラー212に供給する。即ち、水平アップ・サンプラーは要求された数のペルを要求された順序でその中に“引き込む”。ディスプレイラインの終わりにて、チャンネルの一つが最初にデータが無くなり、このことは他のチャンネル(もし有れば)の残りのデータは廃棄されねばならないことを示している。

【0449】VTG333は単純にラスタの間カウントし、出力マルチプレックス371に供給される一連のタイミング信号を生成する。これらの信号の幾つかは出力マルチプレックス371に最終ラスタをどの様に生成するかを報じる内部信号である。他の信号は“外部”信号、例えば同期及びブランキング信号であり、これらはまた出力マルチプレックス371に回路に供給されてデータと同一のクロック数だけ遅延される。

【0450】出力マルチプレックス371ブロックは幾つかの役割を有する。これらの内最も興味深いものは、おそらくデータから二重“インターフェース性”を除去する作業である。水平アップ・サンプラー212から供給されたデータは未だ関連するパリティ(妥当)信号を有している(そして出力マルチプレックスは受け入れ信号を生成する)。マルチプレックスの出力におけるデータは2線式インターフェースを有しておらず、クロック・サイクル毎に1バイトずつ単純にクロック出力される。

【0451】出力マルチプレックス371はまた画面の回到境界を描く役割を担っている。上端および左の境界はVTG333による制御に従って描かれる。VTG333は単純に出力マルチプレックス371に境界色の

ペルを必要な数だけ生成するように命じる。画面の右及び下端においては、出力マルチプレックス371はそれ自身で境界を描く、即ち、マルチプレックス371は画面データが無くなるためそれを行うべきことを知っているのである。

【0452】ディスプレイパイプライン内の最後のブロックは8ビット-16ビット出力モード変換器372である。これは極く単純にフリップフロップ及びマルチプレックスから成る。これは、出力PADそれ自身において実現することが意図されている。これを行うことにより、16ビットバスを引く代わりに単純に8ビットバスを引くことが可能になる。各々のビットは二つの出力パッドに進む。

【0453】水平アップ・サンプラー序文

本発明に依れば、水平アップ・サンプラー212はデコードされた画面をディスプレイラスタに適合するように拡大するためにアップ・サンプリング若しくは補間する機能を行う。

【0454】本発明のアップ・サンプラー212は以下の4つのモードで動作することが可能である。

- 1) 1:1-出力は入力と同一
- 2) 2:1
- 3) 3:2
- 4) 4:3

画面シミュレーションを幾つか行うとともに、考えられる実現費用を考慮した後に、補間を行うために三タップのフィルタを使用することが決定された。

【0455】このフィルタはフィルタは、フィルタ係数の異なる組を用いて各連続した出力が生成されるという意味で多層のフィルタである。位相の数は常にアップ・サンプリング比の分子に等しい。このようにして、4:3アップ・サンプラーは4つの位相を有し、各4番目の出力サンプルが同一のフィルタ係数を用いて生成される。

【0456】アップ・サンプラー212はそれが入力データとして受け入れるより多くの出力データを生成するため、新しい入力サンプルが各クロック周期毎に受け入れられないことは明かである。事実、フィルタが新しい入力を受け入れない位相の数はアップ・サンプリング比の分子と分母の差異に等しい。此の各々(1:1を除く)においてこれは1である。従って、位相の回りの各完全なサイクルにおいて、その位相の内の一つにおいて新しい入力データが受け入れられない。この場合、データは前の位相のデータと同一である。フィルタ係数は、しかしながら、前の位相の係数とは異なっている。

4:3アップ・サンプリング

4:3アップ・サンプリングにおいて、フィルタ係数は表44に示されており、図88はフィルタの動作を示している。出力ペルは本質的には入力ペルの重み付き平均



として形成される。

【0457】

【表44】

位相	C [0]	C [1]	C [2]
0	0	356	0
1	42	220	-6
2	128	128	0
3	-6	220	42

表44 4:3フィルタ係数

新しい入力データは最終位相（位相3）が計算される前に受け入れられることに注意すべきである。

3:2アップ・サンプリング

表45は3:2アップ・サンプリングを示しており、図89はフィルタ動作を図示している。

【0458】

【表45】

位相	C [0]	C [1]	C [2]
0	0	256	0
1	68	194	-6
2	-6	194	68

表45 3:2フィルタ係数

2:1 アップ・サンプリング

同様に、表46は2:1アップ・サンプリングを示しており、図90はそのフィルタリングを示している。

【0459】

【表46】

位相	C [0]	C [1]	C [2]
0	0	256	0
1	0	128	128

表46 2:1フィルタ係数

位相1はフィルタ係数、128、128、0を有するものとしても同様に記述することができることに注意すべきである。これは、フィルタ係数が4:3アップ・サンプリングの位相2のためのフィルタ係数と同一になるという利点がある。これは、しかしながら、“最終の位相を計算している間に新しい入力を受け入れられないという”規則が成り立たないという不利益がある。

境界の効果

画面の端において、画面領域の外側にあるペルから形成された出力ペルを生成する必要がある。この問題を回避するために、エッジペルをピクセルごとに繰り返し、フィルタが画面のエッジであることを認識せずに動作可能であることが必要である。

【0460】三タップフィルタの場合、本発明の場合の如く、画像の左のちょうど1ペルと右の1ペルを繰り返す必要がある。（5タップのフィルタの場合左の2ペルと右の2ペルを必要とするはずである。）これは図91に示されている。概念的には、従って、これを実動化し

た装置は二つの箱から形成されていると見なすことができる。

【0461】DRAMインターフェースは常に16ペル幅の倍数のデータを供給するため、画面が16ペル幅の倍数でない場合にはこの構成は真に正しく動作しないことに注意すべきである。しかしながらこの問題が知られているけれども、我々はそれについては何もするつもりは無い。何れにしても、殆どの画面は16ペル幅の倍数であり、また何れにしても境界効果により影響されるのはラインの最も後のペルのみであるからである。

【0462】このことは図92に示されている。

出力ペルの数

本発明において、アップ・サンプラーは、与えられた数の入力ペルに対して定義された数の出力ペルを生成する。このことは、バーサタイルマシンにとって、画面がラスタに適合するためにアップ・サンプラーの出力において幾つのペルが生成されるかを、よって、幾つのペルを切り落とす（または境界ペルを付加する）必要があるかを決定することを可能にするものであり、特に重要である。

【0463】水平アップ・サンプラーからの最初の有効な出力はアップ・サンプラーに入力される3番目のペルに依って行われなければならない（これは三タップのフィルタであるから）。1ペルが繰り返されるため、これは第2の実際のペルがアップ・サンプラーに入力された時である。最後の有効な出力は、最後の（即ち、繰り返された）入力されたペルに依って可能な出力サンプルの全てが生成されたときに起きる必要がある。多相フィルタの最後の位相が最後から2番目の位相と同一の入力データを使用して計算されるため、このアップ・サンプラーに入る最後のペルが繰り返される結果、一つまたは2つの出力ペルが生成される可能性がある。

【0464】もしこれが行われると、アップ・サンプラーは“q”出力サンプルを生成する。

式1.

$$q = N(pDIVM) + (pREMM)$$

ただしN:Mアップ・サンプラーの場合に“p”入力サンプルに依って。

【0465】例えば、4:3アップ・サンプラーの場合に、表47を以下の様に書くことができる。

【0466】

【表47】

p (入力ペル)	q (出力ペル)
1	1
2	2
3	4
4	5
5	6
6	8

表47 4:3アップ・サンプラーの出力ペル数

## 位置信号

本発明においては、二つの信号がビデオデータとともに転送される。それらはデータが出力ラスタ内での適切な位置に描かれることを出力マルチプレックスが確認することを可能化する。それらは：

・last\_in\_line  
・field\_id

last\_in\_lineは1ペルの間アクティブであり、そのペルがスキャンライン内の最後のペルであることを報知する。

【0467】field\_idはデータがどのフィールドに属するかを示す。“0”は空間的に上のフィールドを表し、“1”は空間的に下のフィールドを表す。この対応関係は境界ライン等の下に成り立ち、デコードされた画像に適用されることに注意すべきである。field\_idペルに対して状態が変化するのが早すぎ、即ち、フィールドの最後から2番目と最後のペルの間で変化する。これにより、フィールドの最後のペルが次のフィールドの最初のペルを待たず判別可能になる。しかしながら、復号がなにかの理由で停止した場合には“次のフィールド”が存在しないかも知れない。field\_id信号は図93に示されている。

【0468】もし正しいフィールドインジケータが必要となときには、それはfield\_idを1ペルの間遅延させることによって得ることができる。これらの信号はディスプレイバンプライン全体においてデータに平行してその動作を行うものであり、多数のフリップフロップを節約することになるため、3つの信号（フィールド信号の最後のペルを許すことによる）ではなく、二つの信号を用いることが重要である。

## 多重化データ

位置信号が多重化データに適用される場合には、注意が必要である。

【0469】データC<sub>lv</sub>、C<sub>rv</sub>の順に多重化されている。本発明において、3つのサンプル(C<sub>lv</sub>、C<sub>r</sub>)同じ時間に存在し、従って、分割不可能にみる筈である。残りのバイト(v)は先行する(C<sub>lv</sub>、C<sub>r</sub>)ペルと以降のC<sub>lv</sub>、C<sub>rv</sub>ペルの間に位置する。その結果、最後のライン内のバイトはC<sub>r</sub>とv内の何れかである。

(3:2のアップ・サンプリングはYペルを奇数個生成するかもしれないことに注意すべきである。)もしラインの最後のバイトがC<sub>r</sub>であれば、ラインの最初のバイトが常にC<sub>lv</sub>であるため、マルチプレックス信号に不連続性が生じることになる：

(C<sub>lv</sub> Y<sub>1</sub> C<sub>r</sub>) (v) (C<sub>lv</sub> Y<sub>1</sub> C<sub>r</sub>) \ (C<sub>lv</sub> Y<sub>1</sub> C<sub>r</sub>) (v) (C<sub>lv</sub> Y<sub>1</sub> C<sub>r</sub>)

## 水平位置合わせ

アップ・サンブラーの入力において、3つの異なるチャンネルが整列する保証はない。

【0470】整列を行うために、本発明においては、水

平アップ・サンブラーと水平整列ブロックの間で”プロトコル”が一致している必要がある。本発明によれば、プロトコルは以下の様に実行される。

・水平ブロックは要求に従って水平アップ・サンブラーにペルを供給する。一つのチャンネルに対してデータが無くなったときには、水平ブロックはこのことをラインの最後のペルをマークする信号を使用してフィルタに報知する。これはペルの繰り返しの場合にのみ起こる。

【0471】・水平アップ・サンブラーは一つのチャンネルから一度最後のペルが供給されると、現ラインにおいてはそのチャンネルから他のペルを要求しないことを保証している。しかしながらフィルタは動作し続け、データが無くなったことをフィルタが知るチャンネルからペルを要求する直前まで他のチャンネルから必要なペルを取り込む。フィルタは出力において生成し得る最後のペルをライン内の最後としてマークする。この時点で、それ自身をリセットしてデータの次のラインのために準備する。

【0472】・水平アップ・サンブラーが、データが既に尽きたチャンネルに対してフィルタがデータを受け入れるを見たとき、それはフィルタが次のラインの最初のペルを求めていることを知る。この時点で、他の二つのチャンネルの残りのペルは全て廃棄される。これらのチャンネルの各々に供給される次のペルがラインの最初のペルとなる。

【0473】二つの別々のブロック（水平整列ブロック及び水平アップ・サンブラーフィルタ）を考慮することが都合が良いけれども、動作の説明上は、この二つは一体に実現されることが好ましい。

## アップ・サンプリング比

アップ・サンプリング比は2ビットの2進数としてフィルタに供給される。フィルタが賢い方法で動作するためには、アップ・サンプリング比はフィールド時間毎に一回アップ・サンブラー自身によってサンプルされることが必要である。次に、この比を供給する回路は次のフィールドに備えるためにその時のフィールドのどの時点においてもサンプリング比を自由に更新することが可能である。

【0474】この比は各フィールドの最初のペルが実際に受け入れられるとともに（前のフィールドの最後のペルの後ではなく）サンプリングされねばならない。この方法によって、リット後（又は復号における何らかの休止の後）の第1のフィールドが正確な比でアップ・サンプリングされる。

## ビデオタイミング生成器

## 序文

この節は本発明によるビデオタイミング発生回路(VTG 333)について記述している。VTGは、種々のアナログビデオ同期信号を生成し、ディスプレイシステムの現在のラスタ位置についての知識を保持することが

その第1の役割である。これにより、V T Gが、アクティブビデオ、境界及びブランキングの信号源を出力用に選択する出力マルチプレクサのための制御信号を提供することが可能になる。アナログ及びデジタル規格の両方がサポートされており、また二つのフレームサイズ(PAL及びNTSC)及び付随する同期の作用がセットアップの際に選択可能である。境界又は切り落とし幅はV T Gにハードウェア入力ロードするトークンの中で指定される。

#### 水平タイミング

水平タイミングパラメータが図94に図示されている。これらは固定のもの(PAL又はNTSCのための)と可変のもの(即ち、指定することが可能ないかなる境界又は切り落としに關係するパラメータ)に分かれる。

【0475】表示されているビデオがインターレースされている性質によって半ラインを基にした計数が必要とされ、種々のタイミングがラインの各半分に對して別々に示される。一つのラインは最初のブランキング期間と、SAVトークンの挿入と、アクティブ期間と、EAVトークンの挿入と、後端のブランキング期間によって構成される。ブランキングラインの間、アクティブ領域は境界及びデータの代わりに挿入されたブランク値を有することになる。

【0476】ライン同期パルスは各ライン(HSYN C)の始めに現れる。ブランキングラインのあるものにおいては、二つの同期パルスが現れ、一つが最初の半ラインの始めに、他方がその後に見える。これらの幅はどちらの垂直領域がアクティブであるか、即ちコライゼーション(等化)又はセレーション(鋸波化)(フィールド同期)に依存している。

【0477】最初の水平ブランキング期間において、ベルは切り落とし値(もし切り落としビットがセットされているとき)に応じて廃棄され、120サイクルの固定期間が先行するラインからRHS切り落としレベルを廃棄するために許されている。次に、現ラインのLHSレベルが廃棄され、アクティブ領域の開始までベルが停止状態とされる。ベルが廃棄されるデータストリームにおいてギャップが存在しないことが重要であり、そうでなければ歪が生じる。

【0478】しかしながら、もし切り落としビットがセットされていないならば、境界Lの期間の間境界値を挿入し、画面幅のデータを挿入し、再び境界をアクティブ領域の終端まで挿入することで境界が構成される。境界R値を計算する必要がないことに注意すべきである。合計の水平境界若しくは切り落とし幅はベル内で指定されている。サンプリングの整合性を維持するためにはLHS境界/切り落とし値は2ベルの倍数でなければならない。その結果として、クックに關しては4の倍数でなければならない。これはベルにおける元の合計境界値から再下位2ビットをマスクして除くことによって達成さ

れる。例えば、指定された境界が91ベルであるとする、と、左の境界は88サイクルの長さになり、画面幅は720-91)\*2サイクルに成る。

【0479】出力(max)に到着するベルのストリームは加算され、32ベルのブロックを与える。この点を考慮すると、サポートされるべきスケール計数と併せて、一ラインについて受け取るべきベルの最大数は832となる。このことは最大切り落とし値は112ベルであり、LHS及びRHSにおいて112サイクルの切り落としとなることを意味している。

#### 垂直タイミングPAL

本発明によるPALに対する垂直タイミングパラメータが図95に示されている。二つのフィールドは、やや異なるタイミングを有するため別々に示されている。アナログパラメータは斜線の領域に示されており、各フィールドに對して同一であり、デジタルパラメータは波形によって示されている。簡単なために、ゼロ境界の場合が示されている。もしゼロ以外の垂直境界が指定されていれば、境界が境界Tの期間挿入され、画面の高さのデータが挿入され、再び境界がアクティブ領域の終わり(固定)まで挿入される。境界T及び画面高さは境界L及び画面幅(水平タイミングおける)と類似した方法で個々に計算される。また再び、上端の境界が2の倍数でなければならないため、初期の境界(境界T)は、今回は半ラインに關して、4の倍数でなければならない。

【0480】MPEGはPALの場合は576ラインのビデオを符号化する一方、アナログ規格は525のみを規定することに注意すべきである。この差異はフィールド毎に576半ラインを出力するためのデータを選択する一方、必要な575ラインについてのみアナログブランキング信号を送出することによって調整される。

#### 垂直タイミングNTSC

次に、本発明によるNTSC垂直タイミングが図96に図示されている。これは少し複雑ではあるけれども、PALタイミングと原理は同様である。MPEGはNTSCについては480ラインのビデオを符号化するけれども、アナログ規格は483を指定している。このことはフレーム毎に境界の3ラインを挿入してギャップを埋めなければならない(フィールド毎に3つの半ライン)ことを意味している。加えて、判断による垂直ブランキングゲインゲータ、V、はアクティブなビデオラインの前にパディングとして追加の境界ラインが挿入されなければならないという方法で指定される。前の節で述べたように、既に示されたラインに加えて、ゼロでない垂直境界が挿入される。更に、垂直切り落としと、どちらの規格においても許されないことに注意すべきである。

【0481】現在のところ、デジタルブランク信号Vに關しては、種々の情報源から矛盾する情報が与えられているためいくらか不確実な点がある。V及びV'により示され、關係する境界選択信号はそれぞれS B及びS

B' である二つの主タイミングの可能性がある。

#### VTG構造

本発明のビデオタイミング発生器は水平及び垂直タイミングの領域に対する別々のマシンから構成される。垂直マシンは水平マシンに対して制御信号を提供し、一方後者は、垂直カウンタに対して半ライン・インクリメント信号を与える。

VTGへの入力は：

- ・クロック及びリセット
- ・NTSCではなくPAL
- ・切り落としの廃棄データビットを有する水平境界値
- ・垂直境界値

出力は：

- ・水平、垂直及び複合の同期及びブランキング信号 s
- ・境界、ブランキング、データの選択信号
- ・切り落としのための廃棄データビット
- ・SAV及びEAV挿入
- ・SAV及びEAVの構成のためのF及びV値
- ・SAV/EAV挿入のための2ビットYUV位置カウンタ
- ・スタート・アップにおける画面の開始を示すための最初のラインビット

同期信号を含めて出力はすべてマルチプレックスブロックに供給され、該同期信号はデータと同期を保つことが可能である。

#### 水平マシン

水平マシンは本質的には図94に示された如く種々のタイミング点の到来を検出するハードウェアを有するカウンタである。カウンタはゼロから半ライン長（これはPAL及びNTSCに対して異なる）に進み、各半ライン毎に繰返される。固定されたタイミング点の各々のためにハード結線された比較器が存在し、これらは規格に従って起動される。加えて、境界値（これはフィールド毎に一度ポーリングされる）のためのレジスタ、画面幅を決定するためのブランキング減算器、境界値からゼロまでカウントダウンする補助カウンタが存在する。この処理は主たる半ラインの生成と平行して行われる。データパスは10ビット幅であり、PAL及びNTSCの両方を実現するためには15のハード結線された比較器が必要とされる。この実施例の構造は図97において、およそのサイズとともに示されている。データパスは360u×330uであると思われる。

【0482】データパスに加えて、本発明のVTG内の殆どの制御ロジックが水平マシンに係る。これはおそらく100-200ゲートに達する。

水平マシンの入力は：

- ・クロック及びリセット
- ・水平境界値及び切り落としビット
- ・ライン、等化、又はフィールド同期インジケータ
- ・NTSCではなくPAL

・垂直ブランク

・垂直境界挿入

水平マシンからの出力は：

- ・水平複合ブランク
- ・データ挿入
- ・境界挿入
- ・ブランク値挿入
- ・入力廃棄
- ・YUVカウントを有するSAV又はEAV挿入
- ・水平同期
- ・複合同期
- ・ラインのスタート
- ・半ラインインクリメント

#### 垂直マシン

垂直データパスは水平データパスと本質的に同一の構造であるが、22のハードウェア比較器（PALに対して8、NTSCに対して14）を有する。主カウンタが各半ライン毎にインクリメントし、各半ラインにおいて半ラインを、また各フィールドにおいて半ラインを交互にカウントする。これもまた10ビット幅である。

【0483】更に、テストのためには半ラインパルス入力をも他のより頻繁なクロックと多重化すること好ましく、これによって垂直マシンを水平マシンとは独立して走らせることが可能である。推測したサイズ360u×420uである。

垂直マシンへの入力は：

- ・クロック及びリセット
- ・NTSCではなくPAL
- ・垂直境界値
- ・半ラインインクリメント

垂直マシンからの出力は：

- ・等化、フィールドまたはライン同期選択
- ・垂直ブランク（アナログ）
- ・垂直同期
- ・SAV/EAV構成のためのF、V及びV'ビット
- ・垂直境界挿入
- ・データ挿入
- ・ブランク値挿入
- ・フレームの開始

#### ハード結線比較器の設計

本発明において、ハード結線された比較器の設計は、ブリッジされるか又はプル・アップを有し、メモリ行デコーダと同様の形で組織された直列のn型トランジスタ列に基づいている。一般的には、これらの比較器は評価が与えられた面積において約8uの高さである。

出力マルチプレックス

本発明の出力マルチプレックスは表示のためにデータを互いに結合する役割を有している。これは、ディスプレイパイプラインの先の部分から到達するデータをVTGから得られたタイミング情報と結合する。

【0484】出力マルチプレックスの他の入力タスクは2線式インターフェースを除くことである。出力マルチプレックスまでの全てのバイラインステージは2線式インターフェースを有しており、実際に出力マルチプレックスの入力に到達するデータは常に到着するのが早すぎ、acceptをローで受け取るため停止状態になる。しかしながら装置の出力においては2線式インターフェースはない。

【0485】上述の2線式インターフェースの除去を達成するために、DRAMインターフェースが水平アップ・サンプラーの出力に到着するデータを停止させることがないようにデータの供給の動態を記憶しておくことが必要である。基本的には、出力マルチプレックスがデータのフィールドを出力するか否かの決定をフィールド単位で行う。出力マルチプレックスは第1のアクティブ・フィールドのラインの開始に近いある時点で決定を行う。もしその入力において待ち状態の有効なデータがあれば(即ち、in\_acceptがローであるとき)、次にそれはデータの出力を開始する。一方、もし、有効なデータが無ければ(例えば、第1の画面がデコードされる前であれば)画面全体に境界色を描く。

【0486】実際には、出力マルチプレックスはデータが正確なフィールドに入力されることを確認しなければならぬためこの操作はやや複雑である。即ち、表示が開始する以前に正確なフィールドに属する有効な待ち状態のデータが無ければならない。もしある時点でデータが有効で無くなれば、或る時間において出力マルチプレックスがディスプレイに描画することができする有効なデータを有すると予想し、(これは実際には起きない)、その後出力マルチプレックスは再び境界色の出力を行い、この出力はフィールドの残りについて続けられる。境界の生成

図98は本発明による画面ディスプレイの左右の境界色の生成を示している。

【0487】図示された如く、VTGは出力マルチプレックス内の境界色選択する信号をアサートすることによって画面の左に境界領域を生成する。しかしながら画面の右側においては、境界色は出力マルチプレックス自身によって生成される。出力マルチプレックスはこれをデータが“無くなった”ことを認識し、境界色の画面の幅の残りを示すことで実行する。

【0488】データが“無くなった”ことは二つの可能な解釈が存在することを理解しなければならない。その一つは水平アップ・サンプラーからの出力データが有効でないことである。しかしながら、これはここで意味されていることではない。この場合には、last\_line信号によってラインの最後であるとしてマークされるベルが出力ストリームに含められた後のデータが無くなったことである。図99は画面の切り落としが

起きた場合に等価な動作を示している。

【0489】図示された如く、VTGは出力マルチプレックスに入力ベルを廃棄するように指令する信号をアサートすることにより出力マルチプレックスに画面の左のベルを切り落とすことを通知する。これが一度発生すると、VTGは出力マルチプレックスが残りのベルの出力を開始すべきであることを通知する。アクティブ・ラインの終わり(即ち、720ベル後)において、VTGは信号のアサートを止め、出力マルチプレックスはその入力におけるデータ内の残りのベルを全て廃棄する。一般に、VTGが切り落としが行われるべきであることを示す時間とアクティブ・ラインの開始の時間との間には間隙(時間の)が存在することに注意すべきである。これはVTGの設計を顕著に簡略化する。出力マルチプレックスは切り落とし信号がアサートされたときにベルを廃棄し、アクティブ・ライン期間の開始まで待機する。

出力マルチプレックス

出力マルチプレックスは種々のデータ源の多重化を制御してCCIR601の8ビット多重化データストリームを形成する。

【0490】タイミング(即ち、何がいつ多重化されるか)は主としてVTGによって制御される。出力マルチプレックスはより高位の問題を処理する。例えば、復号の開始時に、表示すべき画面が無い場合には、出力マルチプレックスは画面全体に境界色を描画する。その結果、最初にデコードされた画面が水平アップ・サンプラーの出力に到着する。一般的に、これは都合よくフィールドの開始においては起きない。出力マルチプレックスはフィールド時間毎に一度“表示可能な有効なデータが存在するか”問い合わせる。もし存在していなければ、次のフィールドが発生するまで待機する(その間の時間に発生した有効なデータ次のフィールドの開始まで待たねばならない。)

出力マルチプレックスはまた、SDRAMインターフェースから到着するデータの正確なフィールドがPAL又はNTSCラスターの正確なフィールドに描画されることを確認する。

【0491】これに加えて、データを扱うために、出力マルチプレックスはビンに出力するための正確な同期及びブランキング信号を選択する。これにより複合エンコーダ、DAC、等の幅広い装置への容易な接続が可能になる。出力マルチプレックスのためのレジスタが表48に示されている。出力マルチプレックスの制御のためのビットは表49に図示されている。

【0492】出力マルチプレックスに連結したMPイレジスタの4つのバイトが有る。

【0493】

【表48】

レジスタ名	サイズ/Dir	リセット状態	記 述
border_cb	8	0 x C 0	境界色の C b 成分
border_r	8	0 x 8 0	境界色の Y 成分
border_cr	8	0 x 4 0	境界色の C r 成分
outaux_ctrl	8	ゼロ	

表 4 8. Outaux レジスタ

【0 4 9 4】

\* \* 【表 4 9】

レジスタ名	ビット	リセット状態	記 述
hs/Cs	0	0	水平同期が hcsync ビンに存在すべきか。抜合同期が hcsync ビンに存在すべきかを制御。0 は抜合同期を選択 1 は水平同期を選択
hcsync_ah	1	0	hcsync ビンのパリティを制御。 0 は active low を選択 1 は active high を選択
vsync_ah	2	0	vsync ビンのパリティを制御。 0 は active low を選択 1 は active high を選択
cblank_ah	3	0	cblank ビンのパリティを制御。 0 は active low を選択 1 は active high を選択
blanking801	4	0	ブランキング中に出力される輝度データの値を制御。 0 は値ゼロを選択 1 は値 0 x 1 0 (16) を選択 C C I R 6 0 1 データに対してはこのピンは 1 にセットされなければならない。
enbl_sav_eav	5	0	出力ストリームにおける S A V 及び E A V 制御語の生成を制御。 0 は S A V 及び E A V を抑制する。この場合、ブランキング値は S A V 及び E A V が生成されていたであろう時間に出力される。 1 は S A V 及び E A V を可能化させる。S A V 及び E A V の間を除いて値ゼロが出力に現れることを防止するためにブランキング 6 0 1 もまた 1 に設定されなければならないことに注意すべきである。 C C I R 6 0 1 データに対してはこのピンは 1 に設定されなければならない。
blank_screen	6	0	1 に設定されたとき、このビットは境界色をスクリーン全体に描画させ、それによってスクリーンがブランクとされる。番号は通常と同様に続けられるけれども、デコードされた画面は見えなくなることに注意すべきである。
vblank	7	—	これは読み出し専用ビット (このビットに書き込まれたデータは無視される)。これは垂直ブランキングを示す。

表 4 9 Outaux\_Ctrl からのビット

a. このビットの設定に拘らず、ブランキングの間色データ (2 8) になる。  
 ータ (C b 及び C r の両方) は 0 x 8 0 (10 進表示で 50 【0 4 9 5】ビデオデコーダの仕様及びその特徴

これまでに述べた詳細な説明に加えて、発明を実施する \* 下の開示内容が提供されている。  
のに適当なビデオコードの好ましい実施例に関して以 \*

- ・MPEG-2MP@ML 2/3及び1/1ブルダウン
- ・単一の16MビットSDRAM ビデオスケーリング
- ・高解像度MPEG-1 SDRAMを含むパワー=2.5W
- ・αビジョン互換 自己構成
- ・自動エラー封じ込め 小ボード面積
- ・チャンネル変更サポート Quiet Pad™ 出力
- ・タイム・スタンプ管理 オン・チップビデオタイミング発生器

本発明は高集積の、使用が容易なMPEG-2ビデオ 10  
コードを含んでいる。それはMPEG-2のメイン・ブ  
ロファイルの要求の全てをメインレベルで完全にサポ  
ートしている。

【0496】本発明のシステムはまた自己構成型（単一  
のピンでPAL及びNTSC動作の間の選択が行われ  
）であり、多くのアプリケーションにおいて外部のソ  
フトウェアのサポート無しでスタートアップ及びビデオ  
の復号の維持が可能である。エラー封じ込め及び復旧は  
完全に自動的である。より要求の多いアプリケーションの  
場合は外部のマイクロプロセッサ上で走るソフトウェア  
によって制御されるアドバンス機能を用いることがで  
きる。

【0497】本発明はそれ自身のマイクロコードをオン  
・チップROMに格納しており、これにより復号が開始  
可能になる前に外部ROMやダウンロードしたマイクロ  
コードを使用する必要を無くしている。図100参照。  
以下に述べた本発明のシステムのより詳細な説明は、系  
統化、明解性及び説明上の便宜のために以下に記述され  
る表題に従って記述されている。

信号、  
レジスタマップ、  
電源、  
ロジック・レベル、  
クロック信号、  
リセット信号、  
コード化データ・インターフェース信号、  
マイクロプロセッサ・インターフェースを介したデータ  
の供給、  
入力モードの切り換え、  
コード化データ受け入れの速度、  
コード化データ・インターフェース・タイミング  
、  
CDクロック、  
ビデオ出力信号、  
ビデオ出力制御レジスタ、  
境界、スケーリングおよび切り落とし、  
ビデオ出力制御レジスタ、  
ビデオ信号タイミング、

MP I 信号、  
MP I の電氣的仕様、  
割り込み、  
ページ・レジスタ、  
SDRAMインターフェース信号、  
SDRAMの構成、  
非JTAGシステムでのJTAGピンの接続、  
ザポートされる命令、  
特性、  
IEEE1149.1準拠のレベル、  
スタートコード検出器レジスタ、  
スタートコードの検出、  
discard\_all機能、  
flag\_picture\_end機能、  
start\_code\_search機能、  
SCDの例-チャンネル変更  
バーサレジスタ、  
エラーコード、  
ユーザーデータの扱い  
システム構成、  
30 信号及びレジスタ、  
電氣的仕様、  
コード化データ・インターフェース、  
ビデオ出力インターフェース、  
マイクロプロセッサ・インターフェース、  
同期DRAMインターフェース、  
JTAGインターフェース、  
スタートコード検出器、  
ビデオ・パーサ、  
タイムスタンプ管理、  
40 アドレス発生器の構成、  
機械的情報  
この節は本発明によって使用されている全ての信号（ビ  
ン）のリストを含んでおり、またマイクロプロセッサ・  
インターフェースを介して使用可能な全てのレジスタの  
リストを含んでいる。（表50及び51参照）  
【0498】  
【表50】

## 信 号

信号名	I/O	ピン 番 号	記 述
CDCLOCK	I	137	コード化データインターフェース。システムにコード化データ又はトークンを供給するために使用される。
CD [7:0]	I	133, 132, 130, 129, 128, 127, 125, 124	
CDEXTN	I	134	
CDVALID	I	123	
CCDACCCEPT	I	122	
EMODE	I	135	
ME [1:0]	I	99, 98	マイクロプロセッサ・インターフェース (MPI)
MR/W	I	97	
MA [5:0]	I	107, 106, 104, 103, 102, 101	
MD [7:0]	O	119, 118, 117, 116, 114, 113, 112, 111	
IRQ	O	96	
DD [16:0]	I/O	36, 35, 33, 32, 30, 29, 27, 26, 21, 20, 18, 17, 15, 14, 12, 11	SDRAM/メモリス
DA [10:0]	O	152, 153, 143, 144, 146, 146, 149, 150, 159, 159, 158, 156, 153	
BS	O	0	
DCKE	O	39	
DCLKOUT	O	38	
DCLKIN	I	23	
DWE	O	9	
DCAS	O	8	
DRA8	O	6	
DSC [1:0]	O	3, 2	
y [7:0]	O	52, 53, 54, 55, 57, 58, 59, 60	ビデオ/メモリス
C [7:0]	O	42, 43, 44, 45, 47, 48, 49, 50	
HCSYNC	O	62	
VSYSN	O	63	
YE	O	64	
CB/CR	O	65	



165

166

V16/8	I	67	
NTSC/PAL	I	68	
CBLANK	O	69	
VTGRESET	I	70	
TCK	I	74	JTAGポート.
TDI	I	73	
TDO	O	72	
TMS	I	75	
TRST	I	79	
SYSCLOCK	I	139	
RESET	I	138	
TIMERSET	I	82	
VCC	-	1, 7, 13, 19, 25, 31, 37, 142, 148, 154, 160	
VDD	-	46, 56, 76, 86, 95, 105, 115, 126, 136	
VDD	-	4, 10, 16, 22, 28, 34, 40, 41, 51, 61, 71, 80, 81, 91, 100, 110, 120, 121, 131, 140, 145, 151, 157	

表50. 信号

【0499】

\* \* 【表51】

信号名	I/O	ピン番号	記 述
TPH0ISH	I	87	
TPH1ISH	I	88	
TSTSTCTRL	I	77	
TLOOP	I	78	通常の動作の間GND 又はVDD に接続 もしPLLSELECT = 0であれば、オンチ ップ位相ロック・ループは機能抑制さ れる。通常動作のためにはPLLSELECT = 1にセットする。
PLLSELECT	I	83	
PLLOCK	O	84	
TDCLK	I	85	

表51. テスト信号

## レジスタ・マップ

本発明のレジスタ・マップは領域に分割されている。始めの32の位置はシステムの通常動作のために要求される。アドレスの5ビットのみが存在する。

【0500】次の32の位置のセットは非省略時SDRAMメモリー・マップを準備するために要求されるアドレス生成回路における位置である。レジスタ・マップの

残りはテスト及び診断のためにのみ用いられるレジスタである。これらはアドレス発生器レジスタの代わりにページ・インされ得る。表52は本発明のレジスタ・マップを示している。

【0501】

【表52】

アドレス (16進)	割り込みサービス	参 照
0x00...0x03	割り込みサービス	
0x04...0x05	入力回路	
0x06...0x07	スタートコード検出器	
0x08...0x0a	タイムスタンプ挿入	
0x0b...0x0f	(未使用)	
0x10...0x17	パーサ	
0x18...0x1c	出力制御	
0x1d	PLL制御	
0x1e	DRAMPAD駆動強度	
0x1f	page_select ね	表3-4
0x20...0x3f	ページング化レジスタアクセス	

表52. 本発明のレジスタ・マップの概観

\* a 通常の動作においては、page\_select は値ゼロを保持しなければならない。この場合には、位置 0x20...0x3f がアドレス生成使用者レジスタを含むことになる。

\* 【0502】表53はページ選択レジスタを示している。

【0503】

\* 【表53】

ページ選択	選 択 さ れ た レジスタ	参 照
0	Addrgen 使用者コンフィギュレーションレジスタ	表3-5
1	組み込み自己テスト及びIDCTテストレジスタ	表3-11 表3-12
2	IK_plusテストレジスタ及びSCDテストレジスタ	表3-13 表3-14
3	パーサテストレジスタ	表3-15
4	フィールド/フレームレジスタ	表3-16
5	BOB テストレジスタ	表3-17
6	付加 BOBテストレジスタ	表3-17
7	Addrgen テストレジスタ	表3-18
8	DRAMIFテストレジスタ	表3-19

表53. ページ選択レジスタ

表54は割り込みサービス領域を示している。

※ 【表54】

【0504】

※

アドレス (16進)	ビット番号	レ ジ ス タ 名	参照ページ
0x00	7	チップ・イベント(chip_event)	
	6	エンド・サーチ・イベント (end_search_event)	
	5	非認識スタート・イベント (unrecognized_start_event)	
	4	フラグ・ピクチャエンド・イベント (flag_picture_end_event)	
	3	パーサ・イベント(parser_event)	
	2		
	1		
	0		

アドレス (16進)	ビット番号	レジスタ名	参照ページ
0x01	7	チップ・マスク (chip_mask)	
	6	エンド・サーチ・マスク (end_search_mask)	
	5	非認識スタート・マスク (unrecognized_start_mask)	
	4	フラグ・ピクチャエンド・マスク (flag_picture_end_mask)	
	3	パーサ・マスク (parser_mask)	
	2		
0x02	1		
	0		
	7	idct 過少イベント (idct_too_few_event)	
	6	idct 過多イベント (idct_too_many_event)	
	5		
	4		
	3		
	2		
	1		
	0	ウォッチドッグ・イベント (watchdog_event)	
0x03	7	idct 過少マスク (idct_too_few_mask)	
	6	idct 過多マスク (idct_too_many_mask)	
	5		
	4		
	3		
	2		
	1		
	0	ウォッチドッグ・マスク (watchdog_mask)	

表54. 割り込みサービス領域

表55は本発明の入力回路レジスタを示している。 40\*【表55】

【0505】

\*

アドレス (16進)	ビット番号	レジスタ名	参照ページ
0x04	7	coded_busy	
	6	enable_gpi_input	
	5	coded_extn	
	4:0	(未使用)	
0x05	7:0	coded_data	

表55. 入力回路レジスタ

表56は本発明のスタートコード検出器を示している

【表56】

【0506】

171

172

アドレス (16進)	ビット番号	レジスタ名	参照ページ
0x06	7	scdp_access	
	6	(未使用)	
	5	discard_extension	
	4	discard_user	
	3	after_search_stop	
	2	flag_picture_end	
	1	after_picture_stop	
	0	after_picture_discard	
0x07	7:3	(未使用)	
	2	discard_all	
	1:0	start_code_search	

表56. スタートコード検出器レジスタ

本発明によれば、表57はタイムスタンプ挿入レジスタ \*【0507】  
を示している。 \*【表57】

アドレス (16進)	ビット番号	レジスタ名	参照ページ
0x08	7:0	ts_high	
0x09	7:0	ts_low	
0x0a	7	vs_valid	
	6	ts_validing	
	5:0	(未使用)	

表 57. タイムスタンプ挿入レジスタ

同様に、表58はビデオパーサレジスタを示している。 ※【表58】

【0508】

※

アドレス (16進)	ビット番号	レジスタ名	参照ページ
0x10	7:0	parser_ctrl0 (実際にはレジスタ・ファイル位置-ビットTBD)	
0x11	7:0	parser_ctrl1 (実際にはレジスタ・ファイル位置-ビットTBD)	
0x12	7:0	parser_error_code (実際にはMSMの一定フィールド)	
0x13	7	parser_access	
	6:0	reg_keyhole_addr	
0x14	7:0	reg_keyhold_data	
0x15	7:0	(未使用)	
0x16	7:0	user_keyhold_addr	
0x17	7:0	user_keyhold_data	

表58. ビデオ・パーサレジスタ

出力制御レジスタは表59に示されている。

【表59】

【0509】

173

174

アドレス (16進)	ビット番号	レジスタ名	参照ページ
0x18	7:0	border_cb	
0x19	7:0	border_y	
0x1a	7:0	border_cr	
0x1b	7 6 5 4 3 2 1	vblank blank_screen embj_sav_eav blanking601 cblank_ah vsyn_ah hcsync_ah hs_not_cs	
0x1c	7:2 1:0	(未使用) vertical upsamle control	

表59. 出力制御レジスタ

テストレジスタ

完全なレジスタ・マップ表60から表69に示されている。

\* 【0510】

【表60】

\*

アドレス (16進)	ビット番号	レジスタ名	参照ページ
P1+00		test_mode	
P1+01...P1+03		(未使用)	
P1+04		misr_mask	
P1+05		(未使用)	
P1+06		misr [1]	
P1+07		misr [0]	
P1+08		psrg_bit_select	
P1+09		psrg_constant	
P1+0a...P1+0c		(未使用)	
P1+0d		psrg [2]	
P1+0e		psrg [1]	
P1+0f		psrg [0]	

表60 組み込み自己テストレジスタ

【0511】

※ ※ 【表61】

アドレス (16進)	ビット番号	レジスタ名	参照ページ
P1+10		idct_elkgen	
P1+11		(未使用)	
P1+12		snp_idct [1]	
P1+13		snp_idct [0]	
P1+14...P1+17		未使用	
P1+18		snp_tram [7]	
P1+19		snp_tram [6]	
P1+1a		snp_tram [5]	
P1+1b		snp_tram [4]	
P1+1c		snp_tram [3]	
P1+1d		snp_tram [2]	
P1+1e		snp_tram [1]	
P1+1f		snp_tram [0]	

表61 IDCTテストレジスタ

【0512】

【表62】

アドレス (16進)	ビット番号	レジスタ名	参照ページ
P2+00		imp_clkgen	
P2+01		(未使用)	
P2+02		snp_quant [1]	
P2+03		snp_quant [0]	
P2+04		(未使用)	
P2+05		snp_inode [1]	
P2+06		snp_inode [1]	
P2+07		snp_inode [0]	
P2+08		snp_quant_ram [3]	
P2+09		snp_quant_ram [2]	
P2+0a		snp_quant_ram [1]	
P2+0b		snp_quant_ram [0]	
P2+0c		iquant_keyhole_data	
P2+0d		iquant_keyhole_addr	
P2+0e...P2+0f		(未使用)	
P2+10		snp_izz_ram [3]	
P2+11		snp_izz_ram [2]	
P2+12		snp_izz_ram [1]	
P2+13		snp_izz_ram [0]	
P2+14		izz_keyhole_data	
P2+15		izz_keyhole_addr	
P2+16...P2+1f		(未使用)	

表62. IN\_plusテストレジスタ

【0513】

\* \* 【表63】

アドレス (16進)	ビット番号	レジスタ名	参照ページ
P2+18		scd_clkgen	
P2+19		(未使用)	
P2+1a		snp_incret [1]	
P2+1b		snp_incret [0]	
P2+1c		snp_cdbin [1]	
P2+1d		snp_cdbin [0]	
P2+1e...P2+1f		(未使用)	

表63 SCDテストレジスタ

【0514】

【表64】

アドレス (16進)	ビット番号	レジスタ名	参照ページ
P3+00		parser_clkgen	
P3+01...P3+02		(未使用)	
P3+03		snr_cdbout [4]	
P3+04		snr_cdbout [3]	
P3+05		snr_cdbout [2]	
P3+06		snr_cdbout [1]	
P3+07		snr_cdbout [0]	
P3+08		(未使用)	
P3+09		snr_aluin [2]	
P3+0a		snr_aluin [1]	
P3+0b		snr_aluin [0]	
P3+0c...P3+0f		(未使用)	
P3+10	7	msn_access	
	6:0	(未使用)	
P3+11	7:3	(未使用)	
	2	msnr_intr_status	
	1	msnr_ss_monitor	
	0	msnr_ss_select	
P3+12	7:4	(未使用)	
	3:0	msn_pc	
P3+13	7:0		
P3+14	7:4	(未使用)	
	3:0	msn_call_return	
P3+15	7:0		
P3+16	7:4	(未使用)	
	3:0	msn_intr_return	
P3+17	7:0		
P3+18		snr_user_ran [7]	
P3+19		snr_user_ran [6]	
P3+1a		snr_user_ran [5]	
P3+1b		snr_user_ran [4]	
P3+1c		snr_user_ran [3]	
P3+1d		snr_user_ran [2]	
P3+1e		snr_user_ran [1]	
P3+1f		snr_user_ran [0]	

表64. パーサテストレジスタ

アドレス (16進)	ビット番号	レジスタ名	参照ページ
P4+00		ff_elkgen	
P4+01		(未使用)	
P4+02		snp_fid_fm [1]	
P4+03		snp_fid_fm [0]	
P4+04		snp_padder_data [1]	
P4+05		snp_padder_data [0]	
P4+06		snp_padder_pf [1]	
P4+07		snp_padder_pf [0]	
P4+08		snp_pf_master [3] (snpseel [3])	
P4+09		snp_pf_master [2] (snpseel [2])	
P4+0a		snp_pf_master [1] (snpseel [1])	
P4+0b		snp_pf_master [0] (snpseel [0])	
P4+0c		snp_pf_slave [3] (snpseel [7])	
P4+0d		snp_pf_slave [2] (snpseel [6])	
P4+0e		snp_pf_slave [1] (snpseel [5])	
P4+0f		snp_pf_slave [0] (snpseel [4])	
P4+10		(未使用)	
P4+11		snp_pf_pipe [2] (snpseel [10])	
P4+12		snp_pf_pipe [1] (snpseel [9])	
P4+13		snp_pf_pipe [0] (snpseel [8])	
P4+14		ff_keyhole_data	
P4+15		ff_keyhole_addr	
P4+16		snp_dec_data [1]	
P4+17		snp_dec_data [0]	
P4+18		snp_ff_ram [7]	
P4+19		snp_ff_ram [6]	
P4+1a		snp_ff_ram [5]	
P4+1b		snp_ff_ram [4]	
P4+1c		snp_ff_ram [3]	
P4+1d		snp_ff_ram [2]	
P4+1e		snp_ff_ram [1]	
P4+1f		snp_ff_ram [0]	

表65. フィールド/フレームテストレジスタ



アドレス (16進)	ビット番号	レジスタ名	参照ページ
P5+00		bob_clkgen	
P5+01		(未使用)	
P5+02		snp_vup_cb [1]	
P5+03		snp_vup_cb [0]	
P5+04		snp_vup_cr [1]	
P5+05		snp_vup_cr [0]	
P5+06		snp_hup_y [1]	
P5+07		snp_hup_y [0]	
P5+08		snp_hup_cb [1]	
P5+09		snp_hup_cb [0]	
P5+0a		snp_hup_cr [1]	
P5+0b		snp_hup_cr [0]	
P5+0c		(未使用)	
P5+0d		snp_outmux [2]	
P5+0e		snp_outmux [1]	
P5+0f		snp_outmux [0]	
P5+10		(未使用)	
P5+11		snp_vtg [2]	
P5+12		snp_vtg [1]	
P5+13		snp_vtg [0]	
P5+14		snp_outface [1]	
P5+15		snp_outface [0]	
P5+16 ... P5+1f		(未使用)	
P6+00 ... P6+0f		snp_vupram_cb1 [7:0] (bobupram)	
P6+08 ... P6+0f		snp_vupram_cb0 [7:0]	
P6+10 ... P6+1f		snp_vupram_cr1 [7:0]	
P6+18 ... P6+1f		snp_vupram_cr0 [7:0]	

表66. BOB テストレジスタ

【0517】

\* \* 【表67】

アドレス (16進)	ビット番号	レジスタ名	参照ページ
P7+0		addrgen_clkgen	
P7+1			
		snoopers	

表67. Addrgen テスト・レジスタ

【0518】

※ ※ 【表68】

アドレス (16進)	ビット番号	レジスタ名	参照ページ
P8+0		dran_clkgen	

表68. DRAMIF テスト・レジスタ

【0519】

【表69】

テスト・レジスタ位置のまとめ

アドレス (16進)	データバス	レジスタ名	位 置
P2+1a ... P2+1b	10	snp_incret [1:0]	チップの入力 (入力回路の前)
P2+1c ... P2+1d	10	snp_edbin [1:0]	edbinの入力
P3+03 ... P3+07	33	snp_edbin [4:0]	edbinの入力
P3+09 ... P3+0b	19	snp_aluin [2:0]	MSM内のALUの入力
P2+05 ... P2+07	19	snp_isode [2:0]	逆モデラーの入力
P2+02 ... P2+03	13	snp_iquant [1:0]	逆量子化器の入力
P1+12 ... P1+13	13	snp_idet [1:0]	IDCTの入力
P4+02 ... P4+03	10	snp_fid_frm [1:0]	7i-8f-7i-8mの入力
P4+04 ... P4+05	10	snp_padder_data [1:0]	padderの変換 データ入力
P4+06 ... P4+07	8	snp_padder_pf [1:0]	padderの予測 フィルタデータ入力
P4+08 ... P4+0b	23	snp_padder_master [3:0]	predfltの マスター入力
P4+0c ... P4+0f	23	snp_padder_master [3:0]	predfltのサブ入力
P4+11 ... P4+13		snp_pf_plpa [2:0]	predfltの半ば
P4+16 ... P4+17	8	snp_dec_data [1:0]	予測加算器の出力
P5+02 ... P5+03	10	snp_vup_cb [1:0]	クロマ・アップ・サンプル Cbの入力
C4P5+04 ... P5+05		snp_vup_cr [1:0]	クロマ・アップ・サンプル 入Crの力
P5+06 ... P5+07	12	snp_hup_y [1:0]	水平アップ・サンプルルyの 入力
P5+08 ... P5+09	10	snp_hup_cb [1:0]	水平アップサンプルCbの 入力
P5+0a ... P5+0b	10	snp_hup_cr [1:0]	水平アップ・サンプルCr の入力
P5+0d ... P5+0f	10+11+12+13 0x1e-7	vig_snp_outaux [2:0]	outauxの入力
P5+11 ... P5+13		snp_vtk_ [2:0]	VTGへの全制即入力
P5+14 ... P5+15	13	snp_outiface [1:0]	8から16変換後の直前、 ピンのための再入力

表69 スノーバ・レジスタ

## 電源供給

本発明は本質的には単一の5V電源からの電力で動作す

\* めに、3.3V電源もまた提供されている。

【0520】

る。しかしながら同期DRAMとの単純な接続を行うた

【表70】

記号	パラメータ	最小	最大	単位
VDD	接地に対する公称5Vの供給電圧	-0.5	6.5	V
VCC	接地に対する公称3Vの供給電圧	-0.5	6.5	V
V <sub>IN</sub>	SDRAMインターフェースピンを除く全てのピンへの入力電圧	接地-0.5	VDD + 0.6	V
V <sub>INDRAM</sub>	全てのSDRAM4777-7-8-10への入力電圧*	接地-0.5	VCC + 0.5	
T <sub>A</sub>	動作温度	-40	+85	℃
T <sub>S</sub>	保存温度	-55	+150	℃

表70. 推奨仕様定格\*

a D [15:0]、DA [11:0]、DCKE、D※ ※CLKOUT、CDLK  
IN、DWZ、DCAS、DRAS、DCS [1:0] 及びTDCLK

b ここで一覧表示された値以上の電圧を加えると装置の永久的な損傷が起きる可能性がある。これは電圧の定格を示すのみであり、これらの条件、またはこの明細書の動作についての節に示された以上の他の条件での装置

の機能的動作は含まれない。絶対最大定格条件下に長時間おくことは信頼性に影響が有り得る。

【0521】

【表71】

記号	パラメータ	最小	最大	単位
VDD	接地に対する公称5V供給電圧	4.75	5.25	V
VCC	接地に対する公称3.3V供給電圧	3.00	3.60	V
GND	接地	0	0	V
T <sub>A</sub>	動作温度	0	70	℃
I <sub>DD</sub>	RMS電力供給電流			mA

表71. DC動作条件

ロジック・レベル

本発明によれば3つの異なる信号インターフェースタイプが実現されている。標準の(5V)TTLレベルがマイクロプロセッサ・インターフェースにより使用される。加えて、5VCMOSレベルがコード化データインターフェース及びビデオ出力インターフェースによつて

\* として使用される。3V LVTTLレベルがSDRAMインターフェースにより使用される。

TTL (5V) レベル

【0522】

【表72】

記号	パラメータ	最小	最大	単位
V	入力論理 "1" の電圧	2.0	VDD + 0.5	V*
V <sub>IL</sub>	入力論理 "0" の電圧	接地-0.5	0.8	V
V <sub>OL</sub>	出力論理 "0" の電圧		0.4	V
V <sub>OLoc</sub>	開放コレクタ出力論理 "0" 電圧		0.4	V*
V <sub>OL</sub>	出力論理 "1" 電圧	2.4		V
I <sub>O</sub>	出力電流	±100		μA*
I <sub>OCC</sub>	開放コレクタ出力電流	4.0	8.0	μA
I <sub>OZ</sub>	出力オフ状態漏れ電流		±20	μA
I <sub>IN</sub>	入力漏れ電流		±10	μA
C <sub>IN</sub>	入力容量		5	pF
C <sub>OUT</sub>	出力/IO容量		5	pF

表72. TTL (5V) DC特性

a 交流入力パラメータは1.4V測定レベルで測定された。

b  $I_{O} \leq I_{OCCmin}$

c これはインターフェースの定常状態駆動能力であり、遷移状態の電流はより大である。

d 信号が送出された(アサートされた)とき開放コレクタI<sub>RQ</sub>出力は100Ωまたはそれ以下のインピーダンスでプルダウンする。

\* CMOS (5V) レベル

CMOS入力に対してはV<sub>IHmin</sub> はVDDのおよそ70%であり、V<sub>ILmax</sub> はVDDのおよそ30%である。表73に示された値はV<sub>IH</sub>及びV<sub>IL</sub>のそれらの動作の極限に対する値である。

【0523】

【表73】

記号	パラメータ	最小	最大	単位
V <sub>IHmin</sub>	入力論理 "1" 電圧	3.68	VDD + 0.5	V
V <sub>ILmax</sub>	入力論理 "0" 電圧	接地-0.5	1.43	V
V <sub>OHmin</sub>	出力論理 "1" 電圧	VDD-0.1		V*
		VDD-0.4		V*
V <sub>OLmax</sub>	出力論理 "0" 電圧		0.1	V*
			0.4	V*
I <sub>IHmax</sub>	入力漏れ電流		±10	μA
C <sub>IHmax</sub>	入力容量		5	pF
C <sub>OUTmax</sub>	出力/IO容量		5	pF

表73. CMOS (5V) DC特性

a  $I_{OH} \leq 1mA$

b  $I_{OH} \leq 4mA$

c  $I_{OI} \leq 1mA$

d  $I_{OI} \leq 5mA$

LVTTL (3.3V) レベル

【0524】

【表74】

記号	パラメータ	最 小	最 大	単位
$V_{IH\text{data}}$	入力論理 "1" 電圧		$V_{CC} + 0.5$	Va
$V_{IL\text{data}}$	入力論理 "0" 電圧	接地-0.5	0.8	V
$V_{OH\text{data}}$	出力論理 "1" 電圧			V
$V_{OL\text{data}}$	出力論理 "0" 電圧			V
$I_{OH\text{data}}$	出力電流	$1 \pm 100$		$\mu A$
$I_{OZ\text{data}}$	出力オフ状態漏れ電流		$\pm 20$	$\mu A$
$I_{IN\text{data}}$	入力漏れ電流		$\pm 10$	$\mu A$
$C_{IN\text{data}}$	入力容量		5	pF
$C_{OUT\text{data}}$	出力/IO容量		5	pF

表74. LVTTTL (3, 3V) DC特性

a 交流入力パラメータはV測定レベルで測定された。

b これはインターフェースの定常状態駆動能力であり、遷移状態の電流はずっと大である。

クロック信号

本発明は殆ど全てのオン・チップ機能のために一つのクロック (SYSCLOCK) を使用している。このクロックはビデオ出力回路によって使用されるため、VTG (ビデオ タイミング発生器) が正確な速度で画面を生成するために27MHzのクロックが使用されることが 20 前提とされている。

【0525】本発明にコード化データを調和させるために第2のクロック (CDCLOCK) を用いても良い。このクロックはSYSCLOCKに同期され、これによ\*

\* 727MHzクロックで動作しない回路 (おそらく、ディスク又はネットワーク・インターフェース回路から導出されるクロック) からデータをこの装置に転送することが可能になる。

【0526】内部的には、本発明は位相ロックド・ループ (PLL) を用いてSDRAMインターフェースを駆動するための高速クロックを導出している。このクロックはSDRAMにDCLKOUTとして出力される。均等なマーク・スペース比を導出するためにオン・チップPLLが使用されている。SYSCLOCKのための要件が図101に示されている。

【0527】

【表75】

番 号	特 性	27MHz		単 位	備考
		最小	最大		
1	クロック期間	37		ns	*
2	クロック・ハイの期間	10		ns	
3	クロック・ローの期間	10		ns	

表75. 入力クロックの要件

a ビデオ規格のライン周波数に適合するためにクロックの許容度及び安定性が充分でなければならないことに注意すべきである。

リセット信号

本発明は3つのリセット信号を使用する。

RESETはメイン・チップリセット信号である。全ての回路がリセットされ

、ここで記述された種々の表に示されたリセット状態を★ ★とる。正確なリセットを保証するためにRESETは電源及びクロックが安定した後に少なくとも4クロックサイクルにわたって出力 (ロー) されねばならな

☆【0528】

い。

☆40

VTGRESETは本発明のビデオタイミング発生器を本発明の他の局面に影響を与えることなくリセットするために用いられる。

TIMERSETは本発明によるタイムスタンプ処理回路によって使用され

る。序文

本発明によれば、コード化データインターフェースは、コード化ビデオデータをシステムに供給するために使用可能な専用のピンの組を提供する。或いは、コード化データはマイクロプロセッサ・インターフェースを介して書き込まれる。この節は、これらの方法の両方について記述している。

【0529】もし専用のピンが使用されていれば、コード化データは単純なバイトのストリーム或いは"トークン"として供給される。トークンの場合は他の種類の情報をコード化データに加えて供給することが可能である。例えば、この機構を用いてタイムスタンプ情報を転送することができる。もしコード化データのためにマイクロプロセッサ・インターフェースが使用されてい

ば、トークンが常に使用される。更に、これは非常に単純である。一度“トークン・ヘッド”が書き込まれて以降のデータがコード化データ（ただ二つのレジスタが書き込まれることが必要とされる）であることが宣言されると、その後、コード化データを単純にレジスタに書き込むことができる。

\* コード化データ・インターフェース信号  
表76に本発明において用いられるコード化データ・インターフェース信号が定義されている。  
【0530】  
【表76】

信号名	タイプ	記述
CD [7:0]	I	コード化データは本発明によれば一度に1バイトずつ供給される。データは CDCLOCK の立ち上がりエッジでサンプリングされる。データは1バイト整列されているものとする。
CDEXTN	I	トークンを転送するためにコード化データインターフェースが使用されているときには、この信号が駆動ビットとなる。この信号は CD [7:0] と同一の時点でサンプリングされる。
CDVALID	I	CDVALID は CD [7:0] と同一の時点でサンプリングされる。これがハイのとき、データは有効であり、コード化データとして使用される。これがローのとき、データは有効でなく、システムにより無視される。
CDACCEPT	O	CDACCEPT はシステムがデータを受け入れる用意ができているか否かを示す。これがハイのとき、期待される通り CDCLOCK の立ち上がりエッジでデータがラッチされる。これがローのとき、システムはデータを受け入れることはできず（おそらく、その内部のバッファが飽和しているためである）、従って、データは再び提供されねばならない。
BMODE	I	この信号がハイであるとき、データはコード化データバイトの単純なストリームであると解釈される（そして CDEXTN は無視される）。これがローであるとき、データはトークンとして解釈される。この信号 CD [7:0] と同一の時点でサンプリングされる。
CDCLOCK	I	このクロックはシステムへのデータの転送を制御するために用いられる。CD [7:0]、CDEXTN、BMODE 及び CDVALID は CDCLOCK の立ち上がりエッジでサンプリングされ、外部回路は同一の時点で CDACCEPT をサンプリングしなければならない。規定（リセット）状態においては、CDCLOCK 及び SYSCLK は同一の信号に接続されねばならないことに注意すべきである。

表76. コード化データ・インターフェース信号

CDVALID 及び CDACCEPT は本発明によればデータの転送を制御するために用いられる。この種の通信制御手順は“2線式”インターフェースと称される。データ転送が行われるためには両方の信号は CDCLOCK の立ち上がりエッジでハイでなければならない。図102はデータ（CD [7:0]、CDEXTN 及び BMODE）と CDVALID 及び CDACCEPT との間の関係を示している。もしデータをコード化データ・インターフェースピンを介して供給すべき場合には、マイクロプロセッサ・インターフェースレジスタ“enable\_mpi\_input”はゼロ（これがそのリセット状態である）でなければならないことに注意すべきである。

バイト・モード

本発明においては、もし BMODE が CDCLOCK の立ち上がりエッジでハイであるとサンプリングされ（更に CDVALID 及び CDACCEPT が共にハイ）であれば、データは単純なコード化データであるとして扱われる。実際には、データは直ちに DATA に組み入れられる。この場合には、CDEXTN は無視される。トークン・モード

もし BMODE が CDCLOCK の立ち上がりエッジでローであるとサンプリングされれば（また CDVALID 及び CDACCEPT が共にハイであれば）、データはトークンとして扱われる。トークンは本発明によって、システム全体においてデータ及び制御信号の流れを制御するために広く使用される。理論的には、コード化データ入力においてどのようなトークンを供給することも可

能である。本発明によれば、全てのトークンは、一連のバイト (CD [7:0]) から成り、その各々には拡張ビット (CDEXTN) が組み合わされている。トークンの最初のバイトはトークンによって運ばれる情報の種類を示している。トークンの最後のバイトはローである拡張ビットによって示される。

【0531】例えば、コード化データはデータトークンを用いて供給される。これは図103に示されている。図示された如く、最初のバイトは0x04 (それがデータトークンであることを示す) である。この情報の後に10はCDEXTNがローであるとサンプルされるまで続くコード化データのバイトが続く。次にサンプリングされるデータは新しいトークンの最初のバイトであると解釈される。(BMODEが未だローであると仮定する)。

【0532】特に有用な他のトークンはFLUSHトーン

7bit (16進)	ビット	Dir/Port	レジスタ名	記 述
04	7	RO/1	coded_busy	このレジスタの状態はシステムが coded_data [7:0] に書き込まれたトークンを受け入れ可能であるか否かを示す。 値1はインターフェースがビジーであり、データを受け入れられないことを示す。 coded_busy = 1のときに使用者が coded_data に書き込みを行うおとしたときの動作は定義されない。
	6	RW/0	enable_mpi_input	コード化データがコード化データポート (0) を介して入力されるか又は MPI (1) を介してシステムに入力されるかを制御する。
	5	RW/x	coded_extn	coded_data に書き込まれたトークンデータの拡張ビット
	4:0	(未使用)		
05	7:0	RW/x	coded_data	トークンデータはこの位置に書き込まれる。

表77. コード化データ入力レジスタ

MPI を介したトークンの書き込み

効率的なデータの転送を可能化するために、コード化データレジスタはメモリマップ内で二つのバイトにグループ化される。8つのデータビット、coded\_data [7:0] が、第1の位置であり、制御レジスタs、coded\_busy、enable\_mpi\_input 及び coded\_extn が第2の位置である。(表56参照)

MPI を介してトークンを入力するように構成した場合には、coded\_data [7:0] に値が書き込まれる毎に現在のトークンは coded\_extn の現在の値によって拡張される。トークンの最後の語が coded\_data [7:0] に書き込まれる前に coded\_extn を0に設定することはソフトウェアの責任である。

【0534】例えば、データトークン coded\_extn

\* クンである。このトークンは“リセット”と同様に作用し、システムを次のビデオストリームに対して準備させるために、一つのビデオストリームの終端の後に用いることができる。FLUSHトークンは図104に示されている。

マイクロプロセッサ・インターフェースを介してのデータの供給

本発明において、トークンはコード化データ入力レジスタにアクセスすることによってマイクロプロセッサ・インターフェース (MPI) を介してシステムに供給される。表77がコード化データ入力レジスタを定義している。

【0533】

【表77】

tn に1を書き込み、次に coded\_data [7:0] に0x04を書き込むことで開始する。この新しいデータ・トークンの開始は次に、処理のためにシステムに渡される。coded\_data [7:0] に新しい8ビットの値が書き込まれる毎に、現在のトークンが拡張される。coded\_extn は現在のトークンを終了するとき (例えば、他のトークンを導入するために) にのみ再びアクセスする必要がある。現在のトークンの最後の語は coded\_extn に0を書き込み、続いて coded\_data [7:0] に現在のトークンの最後の語を書き込むことによって表される。更に、coded\_data [7:0] に書き込む前にはその都度インターフェースがそれ以上のデータを受け入れる用意があるかを確かめるために coded\_busy を調べることが必要である。

入力モードの切り換え

適当な対策が取られるものとして、データ入力モードを動的に変更することが实际的である。一般的に、モードを切り換える前に、何れか一つの経路を通してのトークンの転送が完全に終了していなければならない。これら\*

\* の切り換えモードは表 78 に示されている。

【0535】

【表 78】

前のモード	次のモード	動作
バイト	トークン	オン・チップ回路はバイトモードで供給される最後のバイトをそれ以前に構成していたデータ・トークンの最後のバイトとして使用する（即ち、拡張ビットが 0 にセットされる）。次のトークンを受け入れる前。
	MP I 入力	
トークン	バイト	トークンモードでトークンを供給するオフ回路はトークンを終了すること（即ち、情報の最後のバイトの拡張ビットを 0 に設定することによる）については責任を負わない。バイトモードの選択前。
	MP I 入力	MP I を介した入力へのアクセスはトークンモード内でトークンを供給するオフ・チップ回路がトークンを終了（即ち、情報の最後のバイトの拡張ビットを 0 に設定することによる）するまでは行われない（即ち、 <code>coded_busy</code> が 1 に設定されたままである）。
MP I 入力	バイト	<code>enable_mpl_input</code> が 0 に設定される前に制御ソフトウェアはトークンを終了（即ち、情報の最後のバイトの拡張ビットが 0 に設定されることによる）してなければならない。
	MP I 入力	

表 78. データ入力モードの切り換え

バイトモードで供給される最初のバイトはデータトークンヘッダーがオン・チップで生成されるようにする。バイトモードで更に転送されるバイトは入力モードが変更されるまでこのデータトークンに付加される。MP I レジスタビット `coded_busy` 及び信号 `coded_accept` はどのインターフェースを介してシステムがデータを受け入れようとしているかを示す。これらの信号をただしく観察すればデータが失われていないことを確認することができる。

コード化データ受け入れの速度

本発明の入力回路はトークンをスタートコード検出器に渡す。これは、データトークン内のデータを分析するものであり、その通常の処理速度はクロック（CDCLOCK）の当たり 1 バイトである。しかしながら追加の処理サイクルがときどき必要とされる。例えば、スタート※

※ コードがコード化データの中にあったときである。これが起きた時には、`CDCLOCK` がローになって、データが受け入れられないことを示す。

【0536】その結果、`CDCLOCK` はデータのバイトがシステム供給されるべき速度より高いクロック周波数を有する必要がある。多くの応用においては、`SYSCLOCK` 及び `CDCLOCK` の両方に同一のクロック（典型的には 27 MHz）を用いることが適当である。一例が図 105 に示されている。

コード化データ・インターフェースタイミング

同様に、表 79 は本発明のためのコード化データ・インターフェースタイミングを示している。

【0537】

【表 79】

番 号	特 性	27 MHz		単 位	備考
		最 小	最 大		
1	CDCLOCK サイクル時間	3	7	ns	
2	CDCLOCK ローの時間	1	7	ns	*
3	CDCLOCK ハイの時間	1	7	ns	
4	CDCLOCK 駆動時間		23	ns	†
5	CDCLOCK 保持時間	2		ns	
6	入力信号準備時間	5		ns	
7	入力信号保持時間	0		ns	

表 79. コード化データインターフェースタイミング

a 或状況ではこれらのタイミングは観測される必要はない。

b 最大信号負荷は 20 pF である。

50 【0538】コード化データインターフェースは CMO

S レベルを使用する。

#### CDCLOCK

コード化データインターフェースを通してのデータの転送はビデオデコードクロック (SYSCLOCK) に同期した CDCLOCK によって制御される。この特徴は、システムデコーダがビデオクロックとは異なるクロックで動作することを可能にする点で有用である。

【0539】しかしながら CDCLOCK は本発明においてはスタートコード検出器等の回路に内部的にクロックを供給するためにも用いられている。CDCLOCK は均等なマーク・スペース比を保証する位相ロックループ (PLL) の利益を受けたいため、この点又は図 105 に示されたタイミングパラメータ 2 及び 3 を保証するために外部の回路を用いねばならない。

【0540】CDCLOCK 及び SYSCLOCK が同期している必要のない状況においては、スタートコード検出器等の内部の回路を CDCLOCK よりむしろ PLL から駆動する方が便利である。これにより、外部回路が均等なマーク・スペース比を保証する必要がなくなる。図 106 は CDCLOCK の代わりにスタートコード検出器に供給されるべき、PLL によって生成された均等なマーク・スペース比のクロックを可能にする内部構成を表している。

【0541】もし `un_namcd_register` が 0 (リセット条件) であれば、スタートコード検出器は `is PLL` からクロックを供給される。この場合には、CDCLOCK 及び SYSCLOCK の両方が同一の信号に接続されねばならない。SYSCLOCK のた

めの AC タイミングの要件である。もし `un_namcd_register` が 1 であれば、スタートコード検出器は CDCLOCK を用いてクロックが供給される。この場合には、CDCLOCK は SYSCLOCK に同期される。CDCLOCK は図 105 において指定されるタイミングに従わねばならない。

#### 【0542】序文

本発明のビデオ出力インターフェースは CCIR 勧告 601 及び 656 に準拠したデジタル出力インターフェースを実現している。全ての同期及びブランキング情報が特別な符号語 (SAV 及び EAV) の形で、ビデオ情報と同一のバイト幅のデータのストリームに含まれている。

【0543】加えて、別々の同期及びブランキングピンが設けられており、システムは広い範囲の装置 (ビデオ DAC や NTSC エンコーダ等) に直接接続することが可能である。これらの信号のタイミングは CCIR 勧告 624 に準拠したビデオ信号の生成に適している。ビデオデータは単一バイト幅のバスにおいて時間多重化される。或いは、16 ビット出力モードが設けられ、その場合、輝度データは 1 バイト幅のバスに出力され、一方二つの色差信号は第 2 のバイト幅のバスにおいて時間多重化される。

#### ビデオ出力信号

表 80 は本発明によるビデオ出力インターフェースのための信号を提供する。

#### 【0544】

【表 80】



名 称	タイプ	記 述
Y [7:0]	O	輝度出力データ
C [7:0]	O	Cr/Cb出力データ
HCSYNC	O	水平又は複合同期。マイクログロッセッサレジスタ hesync_ahがどちらの同期がこのピンに出力されるかを制御 する。 レジスタhesync_ahがこの信号の極性を制御する。
VS SYNC	O	垂直同期 レジスタvsync_ahがこの信号の極性を制御する。
CBLANK	O	複合ブランキング レジスタcbblank_ahがこの信号の極性を制御する。
YE	O	SAMPLEの立ち上がりエッジにおいてハイであるとサンプリ ングされた場合、Y (そして16ビットモードではCr又はCb) データが有効である。
CB/CR	O	16ビットモードにおいて、この信号はYEがハイであるとサ ンプリングされたときに、どちらの色成分 (Cr又はCb) が C [7:0] ピンに存在しているかを示す。 8ビットモードでは、この信号はYEがローであるとサンプリ ングされたときに、どちらの色成分 (Cr又はCb) がY [7 0] ピンに存在しているかを示す。
V16/8	I	16又は8ビット出力モードを選択するために使用される。V 16/8がハイであるときに16ビットモードが選択される。 これがローであるときに8ビットモードが選択される。
NTSC/PAL	I	二つの標準ラスタの内のどちらが出力されるべきであるかを 選択する。NTSC/PALがハイのとき、525ラインのラ スタが生成される。これがローのとき、625ラインのラ スタが生成される。 このピンは本発明の動作の他の局面にも影響を与えることに注 意すべきである。
VTPRESET	I	この信号はオン・チップビデオタイミング発生器をリセットす るために送出される。 これは、何等かの外部の制約に対してビデオタイミングをロッ クするために用いられる。

表80: ビデオ出力インターフェース信号

図107は16ビットモードでの出力タイミングを示し  
ている。図108は8ビットモードでの出力タイミ  
ングを示している。  
ビデオ出力制御レジスタ

本発明によるビデオ出力制御レジスタは表81に示され  
ている。  
【0545】  
【表81】

7Pin(16)	ビット	4bit/8bit	レジスタ名	記述
18	7:0	16/0:16	border_cb	境界色のCb成分
19	7:0	16/0:16	border_y	境界色のY成分
1A	7:0	16/0:16	border_cr	境界色のCr成分
1B	7	RO/x	vblank	これは読み出し専用ビット（データこのビットに書き込まれたデータは無視される）である。これは垂直ブランキングを示す。
	6	RW/O	blank_screen	1にセットされたとき、このビットは境界色が画面全体に描画されるようにする。それによって画面が空白にされる。番号は通常と同様に読まれるけれども、デコードされた画面は見えなくされることに注意すべきである。
	5	RW/O	enbl_sav__sav	出力ストリームにおけるSAV及びEAV制御語の生成を制御する。0はSAV及びEAVを抑制し、その場合、ブランキング値が、SAV及びEAVが生成されていたであろう時間に出力される。1はSAV及びEAVを可能化する。blanking01は値ゼロが出力に現れることを避けるために、SAV及びEAVの間以外においては1にセットされねばならないことに注意すべきである。 CCIR601データに対しては、このピンは1にセットされねばならない。

201				202
アドレス(16進)	ビット番号	ビット/セット	レジスタ名	記 述
	4	RW/O	blanking601	<p>ブランキングの間出力される輝度データの値aを制御する。 0は値ゼロを選択する。 1は値0x10(16進)を選択する。 CCIR601データに対しては、このピンは1にセットされねばならない。</p>
1B	3	RW/O		<p>CEBLANKピンの極性を制御する。 0はアクティブ・ローを選択する。 1はアクティブ・ハイを選択する。</p>
	2	RW/O	vsync_ah	<p>VSYNCピンの極性を制御する。 0はアクティブ・ローを選択。 1はアクティブ・ハイを選択。</p>
	1	RW/O	hcsync_ah	<p>HCSYNCピンの極性を制御する。 0はアクティブ・ローを選択。 1はアクティブ・ハイを選択。</p>
	0	RW/O	hs_not_cs	<p>HCSYNCピンに水平同期が存在するか 或いは複合同期が存在するかを制御する。 0は複合同期を選択。 1は水平同期を選択。 (VUPサンプル・モード)</p>

1C

表81: ビデオ出力制御レジスタ

a このビットの設定に拘らず、色データ(Cb及びCrの両方)はブランキングの間0x80(10進表示で 30 128)となる。

境界、スケーリング及び切り落とし(Cropping)

本発明は常に720ペル×480ライン(525ライン・ラスタ)又は576ライン(625ライン・ラスタ)の表示用画面を生成しようとする。本発明はこの領域を埋めるためにデコードされた画面を自動的にスケール(拡大縮小)する。

【0546】限られた数のスケール率のみがサポートされるため、この領域を常に正確に埋めることは不可能である。もしその結果の画面が小さすぎる場合には、デコードされた画面の回りに境界が描画される。この境界はデコードされた画面がスクリーンの中央に位置するように決められている。逆に、もしスケーリングの結果、画面が大きすぎれば、画面は正しく表示されるために切り落とされる。表示された領域はデコードされた画面の中央に位置する。この切り落としはデコードされた画面の

およそ10%より多くが切り落とされないように制限される。もしこの値より多くが失われた場合には、より小さなスケーリング率が用いられる。

【0547】境界色はレジスタborder\_cb、border\_y及びborder\_crに書き込むことによって選択される。装置がリセットされた後であって、画面がデコードされる前には、スクリーン全体が境界色で埋められる。加えて、blank\_screenに書き込むことによってスクリーン全体に境界色を描画することが可能である。これは、例えば、チャンネル変更の間にビデオを隠すために使用することができる。

ビデオ出力特性

特性

図109は、本発明によるビデオ出力インターフェースのタイミングを示している。同様に、表82ビデオ出力インターフェース・タイミングを示している。

【0548】

【表82】

番 号	特 性	27MHz		単 位	備考
		最 小	最 大		
8	出力駆動時間		23	ns	*
9	出力保持時間	2		ns	
10	VTGRESET準備時間	5		ns	・
11	VTGRESET保持時間	0		ns	

表82: ビデオ出力インターフェース・タイミング

a 最大信号負荷は50pFである。

\* 単にリセットが発生する

b このタイミング・パラメータを満足しない場合は、\*

正確なクロックサイクルが不確定になる。VTGRESETには、もしこのタイ

ミング・パラメータが観測されないときのメタスタビリティの問題に対する保護を行うオン・チップ同期装置が設けられている。

※おり、図110がビデオ出力モード信号を示している。

【0550】

【表83】

【0549】表83はビデオ出力モード信号を定義して※

番 号	特 性	27MHz		単 位	備考
		最 小	最 大		
12	セット後の最初のデータの前の準備	5		ns	*

表83: ビデオ出力モード信号

a もしNTSC/PAL又はV16/8がリセット後に状態を変化したときには動作は定義されない。

ビデオ信号タイミング

本発明のビデオタイミングは、その結果生じるビデオ出力が以下のCCIR勧告に準拠するタイミングである。

【0551】

・CCIR勧告601

★・CCIR勧告656

・CCIR勧告624

水平タイミング

水平タイミングは図111に示されている。図中の数は525ラインシステムの場合のSYSCLOCKサイクル数である(625ライン・システムの値は括

★

弧内に示されている)。等価の間、HSYNC信号は62サイクルの間(625

ライン・システムの場合は66サイクルの間)ローである。 ☆【0552】

る。

☆30

フィールド同期の間、HSYNC信号は732サイクルの間(625ライン・

システムの場合は738サイクルの間)ローである。

垂直タイミング

垂直タイミングは図112において525ライン(NTSC)システムについて図示されており、図113において625ライン(PAL)システムについて図示されている。これらの図において、左側下方への数はCCIR勧告656に従ったライン数を与えるものである。右側の二つの列はSAV及びEAVコードにおける"F"及び"V"ビット(CCIR勧告601参照)を与えるものである。

◆【0553】太い実線の、黒ラインの中央の小さな数はデコードされたMPEG画面の論理ライン数を与える。従って、これらは525ライン(NTSC)システムにおいて用いられる480ラインに対しては、0から479の番号が付けられ、625ライン(PAL)システムにおいて使用される576ラインに対しては90から575の番号が付けられている。図114は525ラインシステムについての同期のタイミング及びプランキングピンを示し、図115は625ラインシステ

ムについて示している。HSYNC及びCSYNCの一方のみが出力され(hs

—not—cs参照)、これら信号の各々の極性は反転されても良い(cblank\_ah、等参照)ことに注意すべきである。

VTGリセット状態

本発明において、VTGは525ライン(NTSC)システムに対してはライン4の開始にリセットし、625ライン(PAL)システムに対してはライン1の開始に

リセットする。

【0554】序文

本発明においては標準バイト幅のマイクロプロセッサ・インターフェース(MPI)が使用されている。MPIは種々のデコーダチップクロックと同期して動作する。MPI信号

表84はMPIインターフェース信号を表している。

【0555】

\* \* 【表84】

信号名	タイプ	記 述
ME[1:0]	入力	二つのアクティブ・ロー・チップイネーブル。MPIを介したアクセスを可能化するためには両者はローでなければならない。
MRW	入力	ハイはシステム上のレジスタからの読み出しを示す。ローはシステム上のレジスタへの書き込みを示す。この信号はチップが可能化されている間安定していなければならない。
MA[5:0]	入力	アドレスはチップのレジスタマップにおける位置の内の一つを特定する。この信号はチップが可能化されている間安定していなければならない。
MD[7:0]	出力	8ビット幅データI/Oポート。どちらかのイネーブル信号がハイであれば、これらのピンは高インピーダンスである。
IRQ	出力	アクティブ・ローのオープン・コレクタ割り込み要求信号。

表84. MPIインターフェース信号

MPIの電気的仕様

※ AC特性

DC特性

表85はMPIのための読み出しタイミングを示している。

2. 2. 1. "TTL (5V) レベル" 参照。

20

【0556】図123及び124はそれぞれMPIの読

【0557】

み出し及び書き込みタイミングを図示している。

※ 【表85】

番 号	特 性	最 小	最 大	単 位	備考*
13	ロー期間イネーブル	100		ns	
14	ハイの期間イネーブル	50		ns	
15	チップ・イネーブルのアドレス又はrwの準備	0	ns		
16	チップ・イネーブルからのr/w又はrwの保持	0	ns		
17	出力ターン・オン時間	20		ns	
18	読み出しデータアクセス時間		70	ns	
19	読み出しデータ保持時間	5		ns	
20	読み出しデータ・ターン・オフ時間		20		

表85. マイクロプロセッサ・インターフェース読み出しタイミング

a この例での、サイクルを開始するためのME[0]とそれを終了するためのME[1]の選択は任意に行うことができる。これらの信号は等しいステータス

のものである。

★【0558】同様に、表86はMPIのための書き込みタイミングを示している。

b MD[7:0]の各々における最大の負荷50pFについてアクセス時間が規定されている。負荷が大きければアクセス時間が増大する。

【0559】

★ 【表86】

番 号	特 性	最 小	最 大	単 位	備考
21	書き込みデータ準備時間	15		ns	a
22	書き込みデータ保持時間	0		ns	

表86. マイクロプロセッサ・インターフェース書き込みタイミング

a この例において、サイクルを開始するイネーブル[0]とそれを終了させるイネーブル[1]の選択は任意である。これらの信号は等しいステータスのもの

である。  
割り込み

ブ条件を記述するために用いられる用語である。イベントはエラー状態を表示することが可能であるかまたは使用  
者ソフトウェアに告知することが可能である。

"イベント"は使用者が観測したいと考えるオン・チップ 50

【0560】各割り込み又は“イベント”に関係して二つの単一ビットレジスタが存在する。これらは状態イベントレジスタ及び状態マスクレジスタである。

状態イベントレジスタ

状態イベントレジスタは回路内に生じている状態によってその値が1に設定される1ビットの書き込み/読み出しレジスタである。このレジスタは状態が過渡的にのみ生じる場合であっても1にセットされる。次に、このレジスタは使用者ソフトウェアがそれをリセットするか又はチップ全体がリセットされるまで1に設定されたままであることが保証される。

【0561】・レジスタは1を書き込むことによってゼロにセットされる。

・レジスタへのゼロの書き込みはレジスタを変化されないままにする。

・レジスタは、次にこの状態が観測可能になる前に使用者ソフトウェアによってゼロにセットされねばならない。

・レジスタはリセット時にゼロにリセットされる。

状態マスクレジスタ

状態マスクレジスタは対応する状態イベントレジスタがセットされた時には割り込み要求の生成を可能にする1ビットの読み出し/書き込みレジスタである。もし状態マスクレジスタに1がセットされたときに状態イベントが既にセットされていれば、直ちに割り込み要求が発せられる。

#### IRQ信号

本発明におけるIRQ信号はチップイベントビット及びチップイベントマスクの両方がセットされたときに送出される。IRQ信号はアクティブ・ローの、オ

フ・チップのプル・アップ抵抗を必要とする”オープン” ※ ・コレクタ”出力である

。アクティブ状態のとき、IRQ出力は100Ωまたはそれ以下インピーダンス

によってプル・ダウンされる。殆どの応用において、およそ4kΩのプル・アップ抵抗が適当である。

ページレジスタ

本発明により必要とされるレジスタアドレス信号の数を減少させるために、64以上のレジスタをアドレス指定することを可能にするためにページ・レジスタが用いられる。このページ・レジスタは0x1fの位置に有る。レジスタ位置0x00から0x1fはページ・レジスタの内容により影響されず、常にレジスタマップに存在する。0x20から0x3fの位置のレジスタはページ・レジスタに依存している。

【0564】通常の装置動作に要求されるページングさ

※ 【0562】・値1が割り込みを可能化する。

・レジスタはリセット時にゼロにクリアされる。

他に規定されない限り、ブロックは割り込み要求を生成した後に動作を停止し、状態イベントレジスタ又は条件マスクレジスタのどちらかがクリアされた後すぐに再スタートする。

イベント及びマスクビット

本発明において、イベントビット及びマスクビットは常にレジスタマップ内の連続したバイト(表55参照)における対応するビット位置に組合わされる。これにより、どのイベントが割り込みを生成したかを確かめるために、割り込みサービス・ソフトウェアがマスクレジスタから読み出された値をイベントレジスタ内の値のためのマスクとして使用することが可能になる。

チップ・イベント及びマスク

本発明はチップ上のイベントのアクティビティを要約する単一の”大域的”イベントビットを有する。チップイベント・レジスタはそれらのマスクビットにおいて1を有する全てのオン・チップイベントのORを示す。

20 【0563】チップマスクビットの1はチップが割り込みを生成すること許す。チップマスクビットの0割り込み要求を生成するなどのオン・チップイベントも示している。チップイベントに1又は0を書き込んでも何も効果を生じない。それは全てのイベント(それらのマスクビットにおける1でイネーブルされた)がクリアされるときにのみクリアする。

本発明におけるIRQ信号はチップイベントビット及びチップイベントマスク

の両方がセットされたときに送出される。IRQ信号はアクティブ・ローの、オ

※ ・コレクタ”出力である

。アクティブ状態のとき、IRQ出力は100Ωまたはそれ以下インピーダンス

れたレジスタは存在しない。ページングされたレジスタは、最後に、テストの目的のためにのみ用いられる。本

発明において、ページ・レジスタは値ゼロにリセットされる。使用者は他の値がこのレジスタに書き込まれないことを確実にしなければならない。

序文

SDRAMインターフェース信号

40 表87はSDRAMインターフェース信号を図示している。

【0565】

【表87】

信号名	タイプ	記 述
DD [15:0]	I/O	データピン
DA [10:0]	O	アドレス ピン
BS	O	バンク選択 これはしばしば16MビットSDRAM部品においてA [11] と表示される。
DCKE	I	クロック・イネーブル
DCLKOUT	O	SDRAMクロック出力
DCLKIN	I	DCLKOUTに接続
DWE	O	書き込みイネーブル
DCAS	O	列アドレス
DRAS	O	行アドレス
DCS [1:0]	O	チップ選択 DCS [0] はSDRAMの最初の“バンク”を選択する。もし第2の“バンク”が使用されていれば (SDRAMの構成1及び2を参照)、DCS [1] もまた使用される。

表87. SDRAMインターフェース信号

SDRAMの構成

\* 【0566】

表88はSDRAMの構成を示している。

\* 【表88】

構 成	SDRAMバンク	合計DRAM	組 成
0	1	16Mビット	16Mビット、1M×16ビット
1	2	20Mビット	16Mビット、1M×16ビット 4Mビット、256k×16ビット
2	2	32Mビット	16Mビット、1M×16ビット 16Mビット、1M×16ビット
3	2	32Mビット 16Mビット	16Mビット、2M×8ビット 2M×8ビット

表88. SDRAMの構成

構成ゼロ

構成ゼロのSDRAMの接続については図116参照

図117は一つのSDRAMの接続のための構成を図示している。同様に、図118及び図119は2つ又は3つのSDRAMの接続の構成をそれぞれ示している。

【0567】序文

本発明によるシステムは、現在IEEE規格1149.1に採用されているジョイント・テスト・アクション・グループ (JTAG) の“標準テスト・アクセスポート及び境界スキャン・アーキテクチャ”を完全にサポートしている。全てのJTAGの動作はテスト・アクセスポート (TAP) を介して実行され、後者は5つのピンから構成される。TRESET (テストリセット) ピンがJTAG回路をリセットし、テストモードにおいて装置の電源が立ち上がらないことを確実にしている。TCK (テスト・クロック) ピンが直列テストパターンをTD

I (テストデータ入力) ピンに入力しTDO (テストデータ出力) ピンから出力するためにクロックを供給することに用いられる。更に、JTAG回路の動作モードが適当なビットシーケンスをTMS (テストモード選択) ピンにクロックすることによって設定される。

【0568】JTAG規格はチップ製造者の選択に従って追加の特徴を提供することが可能のように拡張可能である。本発明によれば、3つのJTAGの必須の命令を含む9つのユーザ命令が有る。追加の命令は有る程度の内部装置のテストを可能にし、追加の外部テストを行う柔軟性を提供するものである。例えば、単純なJTAGシーケンスにより全ての装置出力を浮遊させることができる。表89参照。

非JTAGシステムにおけるJTAGピンの接続

【0569】

【表89】

信号	方向	記 述
TRST	入力	このピンは内部のプル・アップを有しているけれども、JTAGの機能を使用していないときであっても電源立ち上げ時にはローでなければならない。これはTRSTをチップリセットピンRESETと共通接続することによって達成することができる。
TDI	入力	これらのピンは内部のプル・アップを有し、JTAG回路が使用されていなければ、接続されないままで良い。
TMS		
TCK	入力	このピンはプル・アップを有しておらず、JTAG回路が使用されていなければ接続されなければならない。
TDO	出力	JTAGスキャン動作の除いては高インピーダンス。もしJTAGが使用されていなければ、このピンは接続されないままで良い。

表89 JTAG入力の接続方法

サポートされる命令

\* 【0570】

この節は本発明のこの実現例でサポートされる命令につ

【表90】

いて記述している。表90、91、及び92参照。 \*

命 令	記 述
EXTEST	これは最も基本的な命令である。これは境界スキャン・チェーンからPCBにデータを供給し、その応答を捕捉する。これは予め定義された命令レジスタにおける全0の命令コードを有する。
SAMPLE/ PRELOAD	この命令は境界スキャン・チェーンがスイッチ・インされることなく、即ち、システム動作に対して透明に装置のピンから並列にロードされ、シフトされることを許容する。これによって、装置のピンの状態の“スナップショット”を取ることができ（準安定状態を回避するために外部クロック制御は必要とされる）、或いはEXTESTモードに切り換える前に境界スキャン・チェーンをプリ・ロードすることができる。SAMPLE/PRELOADのための命令コードは製造者によって選択可能である。
BYPASS	この命令は境界スキャン・チェーンをバイパスするために1ビットバイパス・レジスタを選択し、それによってPCB上の他の装置へのアクセスに必要とされるビット・ストリームの長さを減少させる。この命令コードは予め全1として定義されている。

表90. 必須命令

【0571】

※ ※ 【表91】

命 令	記 述
INTEST	これはEXTESTの逆* を実行する。即ち境界スキャン・チェーンからチップコアにデータを供給し、その応答を捕捉する。命令コードは我々が選択可能である。この機能を役立てるために適当なテストを案出することはユーザに任されている。

表91. サポートされる補助命令

以下の補助JTAG命令はサポートされていない。

2) RUNBIST

【0572】

【0573】

1) IDCODE

【表92】



命 令	記 述
FLOATBS	この命令は境界スキャンレジスタを、全ての開放ドレイン・セルにおいて '1' を含み、他の全てにおいて '0' を含むようにプリセットする。システム動作に影響されない。出力セルにおける '0' は出力を浮動させるため、これは全ての出力を不可能化させる (PCBのテストのための一般的な要件である) 迅速な方法である。出力は、境界スキャン・チェインをスイッチ・インさせる命令、例えばEXTESTがロードされるまで浮動しない。(もしFLOATBSが境界スキャン・チェイン自身をスイッチ・インしたとすると、UPDATE_DR状態まで不明なデータがピンから出力される。)
INEXTEST	INTEST及びEXTESTの組み合わせを行う。何れにしても独立した機能があるため、恐らく非常に役に立つことはないであろう。あるユーザにとってはより高速のPCB/チップの複合テストを案出することが可能になる。多くのJTAG装置が別々のモードよりはるこの組み合わせモードを使用している。
SETBYP	バイパス・レジスタをTDIとTDOの間で選択するけれども、境界スキャン・チェインをスイッチ・インする。これにより、PCBテストが一定のパターンを装置のピンに準備するとともに、第1の装置を再ロードする必要無しに他の装置のピンにアクセスすることを可能にする。テキサス・インスツルメント社の「スコープ」JTAG装置における同一の機能と名称は同じである。
SHIFTBN	SAMPLE/PRELOADと同様であるが、SAMPLE動作は無い。現在の境界スキャンの内容を上書きすることなしにいくらかシフトすることを可能にする。T. I. はその装置「スコープ」においてこの命令を設けているけれども、READBN或いはRBRNMなどの種々の呼び方をしており、それらは何れも非常に直感的な名前ではない。
SHIFTBT	境界スキャン・チェインがスイッチ・インされることを除いてSHIFTBNと同様である。可能性としては、JTAG装置の間で外部接続された小論理ビットのためにPCBテストパターンを最適化するために使用できる点でSHIFTBNよりは有用である。例えば、チェインの遠い端近傍の2入力ゲートのために、幾つかのテスト・パターンを境界スキャン・チェイン内で待機させ、順次供給することができる。EXTESTは反対に、各スキャンサイクルで境界スキャンの内容を上書きする。

表92. 追加の公開命令

命令コードの割当

全体で14の定義された命令が存在する。よって、2つの割り当られない命令を含む4ビット長の命令レジスタが存在する。割り当られない命令はIEEE1149.1に従ったBYPASS命令のエリアスである。

【0574】命令及びそれらのコードの完全なリストが表93に示されている。

【0575】

【表93】

コード	命 令	レジスタ	信号捕捉	B/スキャン クラス
0000	EXTEST	B/スキャン	入力パッド/ O's	ステイビリティ 必須
0001	SAMPLE/ PRELOAD	B/スキャン	全パッド	透過 必須
0010	INTTEST	B/スキャン	O' 出力パッド	ステイビリティ 推奨
0011	FLOATES	B/スキャン	O's	透過 公開
0100	SHIFTBT	B/スキャン	無変化	ステイビリティ 公開
0101	SHIFTBN	B/スキャン	無変化	透過 公開
0110	INTEST	B/スキャン	全パッド	ステイビリティ 公開
0111	指定無し	バイパス	0	透過 予約
1000	専 有			
1001	専 有			
1010	SPDATAT	スキャンデータ	内 部	ステイビリティ 専有
1011	SPDATAN	スキャンデータ	内部レジスタ	透過 専有
1100	SETBYP	バイパス	0	ステイビリティ 公開
1101	指定無し	バイパス	0	透過 予約
1110	BYPASS	バイパス	0	透過 公開
1111	BYPASS	バイパス	0	透過 必須

表93. JTAG 命令コード

IEEE規則1149.1準拠レベル

\*【0576】

以下の点に注意する必要があるけれども、すべての規則

【表94】

が堅持されている。

\*

規 則	記 述
3. 1. 1 (b)	TRSTピンが提供される。
3. 5. 1 (b)	全ての公開命令に対して保証される。(IEEE1149.1. 5. 2. 1 (c) 参照)
5. 2. 1 (c)	全ての公開命令に対して保証される。ある専有命令については、TDOピンが幅足—DR、EXIT1—及びボーズ—DRの内の何れかの状態の間アクティブであり得る。
5. 3. 1 (a)	電源投入リセットがTRSTピンを使用して行われる。
6. 2. 1 (e, f)	バイパス命令のためのコードが Test-Logic-Reset 状態においてロードされる。
7. 1. 1 (d)	割り当てられない命令コードはバイパスと同等である。
7. 2. 1 (c)	装置IDレジスタ無し。
7. 8. 1 (b)	単一ステップ動作はシステムクロックの外部制御を必要とする。
7. 9. 1 (…)	RUNBIST機能は無い。
7. 11. 1 (…)	IDCODE命令は無い。
7. 12. 1 (…)	USERCODE命令は無い。
8. 1. 1 (b)	装置識別レジスタは無い。
8. 2. 1 (c)	全ての公開命令について保証されている。専有命令コードがロードされている間、TDIからTDOへのパスの見かけの長さは特定の状況で変化する可能性が有る。
8. 3. 1 (d-f)	全ての公開命令に対して保証される。専有命令コードがロードされている間、データがTCKの立ち上がりエッジ以外の時間にロードされる可能性が有る。
10. 4. 1 (e)	INTTESTの間、システムクロックピンは外部から制御されねばならない。
10. 6. 1 (c)	INTTESTの間、出力ピンはTDIを介してシフトインされたデータにより制御される。

表94. JTAG規則

推奨項目

【0577】

\*【表95】

\*

推奨項目	記 述
3. 2. 1 (b)	TCCKは高インピーダンスCMOS入力である。
3. 3. 1 (c)	TMSは高インピーダンスのプルアップを有する。
3. 6. 1 (d)	(チップの使用に適用される。)
3. 7. 1 (a)	(チップの使用に適用される。)
6. 1. 1 (e)	SAMPLE/PRELOAD命令コードはCapture-IRの間にロードされる。
7. 2. 1 (f)	INTEST命令がサポートされる。
7. 7. 1 (g)	EXTTESTの間システム出力ピンにゼロがロードされる。
7. 7. 2 (h)	全てのシステム出力が高インピーダンスに設定され得る。
7. 8. 1 (f)	INTESTの間システム入力ピンにゼロが設定される。
8. 1. 1 (i)	装置設計に特有のテストデータレジスタは公開アクセス可能ではない。

表95. 満足された推奨項目

【0578】

※ ※【表96】

推奨項目	記 述
10. 4. 1 (f)	EXTTESTの間、システムクロックピンからオン・チップロジックに送り込まれる信号は外部から供給される信号である。

表96. 実現されない推奨項目

許可

【0579】

★【表97】

★

許 可	記 述
3. 2. 1 (c)	全ての公開命令に対して保証されている。
6. 1. 1 (f)	装置設計に特有の情報を捕捉するためには命令レジスタは使用されない。
7. 2. 1 (g)	いくつかの追加の公開命令が提供されている。
7. 3. 1 (a)	いくつかの専用命令コードが割り当てられている。
7. 3. 1 (c)	(Rule?) からの命令コードが文書化されている。
7. 4. 1 (f)	追加のコードがBYPASSと同様に動作する。
10. 1. 1 (i)	各出力ピンがそれぞれの3値制御を有する。
10. 3. 1 (b)	並列ラッチが提供されている。
10. 3. 1 (i, j)	EXTTESTの間、入力ピンはTDIを介してシフト・インされたデータにより制御される。
10. 6. 1 (d, e)	Test-Logic-Reset 状態において3値セルは非動作状態に強制されない。

表97. 満足された許可

序文

本発明によれば、スタートコード検出器(SCD)は、コード化データストリーム内のスタートコードを検出する役割を担っている。該検出器はこれをシステムによる更に内部の処理のためのトークンに変換する。この処理に加えて、例えばチャンネル変更をサポートする一連

の特徴が存在する。

スタートコード検出器レジスタ

表98は本発明のスタートコード検出器のためのレジスタ図示している。

【0580】

【表98】

アドレス(16進)	ビット番号	Dir/Write	レジスタ名	記述
06	7	RW/0	sdp_access	このビットはレジスタ位置0x07内の値が高精度で書き込まれるためにそれ以前に1に設定されねばならない。これはSCDによるデータの処理を中止させ、マイクロプロセッサアクセスとSCDによるレジスタ自身を修正しようとする動作が競合しないようにする。sdp_accessに一旦値が書き込まれると、マイクロプロセッサはsdp_accessをポーリングし、それが1を読み返すまで待たねばならない。位置0x07に必要とされるアクセスが行われると、SCDがデータの処理を続けることを可能にするためにsdp_accessに値0が書き込まねばならない。
	6	(未使用)		
	5	RW/1	discard_	discard_extension が1のとき、extension MPEG-2MP@MLとして認識されない拡張データはスタートコード検出器において廃棄される。これが0のとき、かかる拡張データはコード化データバッファを介してパーサに渡される。標準のマイクロコードの場合、discard_extension を0に設定する点は存在しない。

221

222

7Fh(16h)	ビット番号	Dir/Type	レジスタ名	記 述
	4	RW/1	discard_user	discard_userが1のとき、いかなるユーザーデータもスタートコード検出器において廃棄される。これが0のとき、ユーザーデータはコード化データバッファを介してパーサに渡される。パーサにおいて少量のユーザーデータを扱う機能が存在するけれども、discard_userが0に設定されている場合は注意を払う必要が有る。システムは任意の量のユーザーデータを扱うことはできないことに注意すべきである。
	3	RW/0	after_search_stop	start_code_search機能と共に使用される。
	2	RW/0	flag_picture_end	これはflag_picture_end 機能を可能化するために1に設定される。
	1	RW/0	after_picture_stop	flag_picture_end 機能と共に使用される。
07	0	RW/0	after_picture_discard	flag_picture_end 機能と共に使用される。
	7:3	--	(未使用)	
	2	RW/0	discard_all	これはdiscard_all 機能を可能化するために1に設定される。
	1:0	RW/0	start_code_search	このレジスタのゼロ以外の値はstart_code_search機能を可能化する。頁84の8、5参照
00	7	--	(スタートコード検出器とは関連していない。)	
	6	RW/0	and_search_event	このビットはstart_code_searchが満足された時はいつも設定される。もし end_search_maskも1に設定されていれば割り込みが生成される。*

7Fh(16h)	ビット番号	Dir/Type	レジスタ名	記 述
01	5	RW/0	unrecognized_start_event	このビットは認識されないスタートコードが検出された時にいつも設定される。もし unrecognized_start_maskも1に設定されていれば、割り込みが生成される。
	4	RW/0	flag_picture_end_event	このビットは画面の終了が検出された、flag_picture_end が1の時はいつも設定される。もし flag_picture_end_maskも1に設定されていれば、割り込みが生成される。頁82の8、4参照。
	3:0	--	(スタートコード検出器に関連していない。)	
	7	--	(スタートコード検出器に関連していない。)	
	6	RW/0	end_search_mask	上記のend_search_event 参照。上記のunrecognized_start_event 参照。
	5	RW/0	unrecognized_start_mask	上記のunrecognized_start_event 参照。
	4	RW/0	flag_picture_end_mask	上記のflag_picture_end_event 参照。
	3:0	--	(スタートコード検出器に関連していない。)	

表9B: スタートコード検出器レジスタ

a イベントビット単純なR/Wレジスタビットでは無 50 い。

223

b 全ての割り込みは chip\_mask が 1 に設定される  
ことが条件である。スタートコードの検出  
本発明のスタートコード検出器は正確に整列されたスタート  
コードのみを検出する。

【0581】本発明はビデオスタートコードのみを扱う。  
認識されないスタートコードは検出され、unrecognized\_start\_code イベントを  
起こさせる。認識されないスタートコードはシステムスタート  
コード (0xb9 から 0xff までの値)、予約済み  
スタートコード (0xb0、0xb1、及び 0xb  
6) 及び sequence\_error\_code (0  
xb4) である。

discard\_all 機能  
discard\_all 機能はシステムに入る全てのデータ  
を廃棄するために使用することができる。レジスタ  
discard\_all を 1 に設定することによって  
discard\_all 機能を“手動で”選択することが  
できる。しかしながら scdp\_access が最初に  
1 に設定され、それが 1 を読み返すまでポーリングされ  
ることが必要である。一般的に、flag\_picture\_end  
機能の一部分として自動的にこのモ  
ードに入る動作が典型である。

【0582】本発明においては、discard\_all  
に値 0 が書き込まれるか FLUSH トークンに出合  
うかするまでは全てのデータが廃棄され続ける。discard\_all  
をリセットする FLUSH トークンはト  
ークンのストリームから削除され、パーサ又は回路の以  
降のいかなるブロックにも影響を与えないことに注意す  
べきである。

flag\_picture\_end 機能  
本発明によれば、flag\_picture\_end 機能\*

スタートコード・サーチ	サーチを終了させるスタートコード
0	(通常動作では無し)
1	picture_start_code、group_start_code and sequence_start_code
2	group_start_code 及び sequence_start_code
3	sequence_start_code

表 99. スタートコード・サーチモード

サーチ・モードに入るためには start\_code\_  
search にゼロ以外の値を書き込む。次にスタート  
コード検出器は、表 99 によって示された適当なスタート  
コードをサーチする。全てのデータ及びトークンはサ  
ーチが実行している間は廃棄される。適当なスタート  
コードの一つに遭遇したときにサーチは終了する。start\_code\_search  
がゼロに設定され、補助  
として割り込みが生成されても良い。

【0586】FLUSH トークンは表示されたスタート  
コードの一つに遭遇したかのようにしてサーチを終  
了することに注意すべきである。しかしながら FLUSH  
トークンが discard\_all 機能を終了させて

224

\* 能はシステムへのデータのフローを停止させる前に画面  
の終了まで待つことによって、復号をきれいに終了させ  
ること意図したものである。従って、パーサは不完全な  
画面を見ることは無い。

【0583】図 120 は flag\_picture\_end 機能  
をフローチャートとして図示している。図示され  
た如く、画面の終端が検出されたときに割り込み (flag\_picture\_end\_event) を生成  
することができる。これは割り込みが処理される迄は S  
CD のデータの処理を中止させる。或いは、SCD は動  
作を続けることが許されても良い。

【0584】もし after\_picture\_discard  
が 1 に設定されていれば、画面の終端が検出  
された後は、全ての以降のデータが廃棄される。これは  
チャンネル変更の前に“航行”中の 1 つのチャンネルか  
らの後続データをシステムデマルチプレクサにおいて廃  
棄するのに最も有用である。この実施例において、start\_code\_search  
機能が flag\_picture\_end 機能に対して優先することに注意  
すべきである。この方法によって、start\_code\_  
search によって廃棄されているデータが画面  
の終端に達したか否かを決定するために調べられること  
が無い。

start\_code\_search 機能  
本発明において、SCD は特定のタイプのスタートコード  
をサーチすることに設定可能である。これは例えば、  
チャンネル変更後に復号の開始の前にシーケンススタート  
コードをサーチするために使用される。

【0585】  
【表 99】

いる特別な場合には、サーチは終了されない。更にこれ  
は、FLUSH トークンに遭遇したときに、discard\_all  
と予め選択されたサーチ・モードとの間で  
直接移動することを可能にする。

【0587】図 121 は本発明による start\_code\_  
search 機能をフローチャートとして図示し  
ている。

SCD の例—チャンネル変更  
本発明における SCD 機能を使用した一例がチャンネル  
変更動作を行う以下の動作シーケンスにおいて示されて  
いる。

1) 制御マイクロプロセッサがチャンネル変更の必要

(おそらく、信号遠隔制御ユニットからの信号に応答して)を認識する。マイクロプロセッサは SCD の flag\_picture\_end 機能を;

・ 1 を flag\_picture\_end に、  
・ 1 を after\_picture\_discard に、

・ 1 を flag\_picture\_end\_mask に、書き込むことによって使用する。

2) スタートコード検出器が現在の画面の終端を検出すると、直ちに以降の全てのデータを廃棄し始める。マイクロプロセッサには割り込みが行われ、割り込みの原因は flag\_picture\_end\_event であると決定する。マイクロプロセッサはまず、

・ 3 (sequence\_start をサーチ) を start\_code\_search に、次に

・ 1 を flag\_picture\_end\_event (イベントをクリアするため)に書き込むことによって新しいチャンネルのためのスタートコード検出器を用意する。

3) 次に、マイクロプロセッサはチューナを再同調させて新しいチャンネルを選択させる。

4) 旧チャンネルからの最後のデータがシステムに転送された後に、(そして新しいチャンネルの最初のデータの前に) FLUSH トークンが挿入される。(或いは、値 0 が discard\_all に書き込まれる。)従って、スタートコード検出器はデータ(旧チャンネルからの)の廃棄を止め、(新しいチャンネルからのデータに対して)シーケンススタートコードのサーチを始め

る。

5) 一度シーケンススタートコードが検出されると、スタートコード検出器はデータの廃棄を止め、通常の復号に復帰する。

【0588】序文

本発明によれば、ビデオパーサは、ビデオデータストリームの復号を担当する。これはマイクロプログラムされたプロセッサとして実現されている。通常のイベントの進行においては、ビデオパーサと交互作用する必要は殆ど無く、多くの単純なアプリケーションは単にビデオパーサにそのビデオを復号する処理を行わせるせる。

【0589】しかしながら、ビデオパーサは例えばビットストリームエラー等の通常でないイベントや予期しないイベントを検出したときに制御用マイクロプロセッサにそれを報知することが可能である。全ての場合において、マイクロコードはエラーから復帰する(そしてエラー随す)ためのコードを含んでおり、ビットストリームエラーを無視しても安全である。しかしながら、ビットストリームエラーが発生しているという知識は診断のために有用である。

【0590】更に、タイムスタンプ管理内のある部分はパーサのマイクロコード・プロセッサによって処理される。これらは第10章において説明されている。

パーサレジスタ

パーサによって使用されるレジスタは表100に示された如くである。

【0591】

【表100】

227

228

アドレス(16進)	ビット幅	Dir/Type	レジスタ	記 述
10	7:1	RW	(parser_ctrl)	何も機能は割当てられていない。
	0	RW	parser_continue	現在の動作を終らせるべきか通常の復号に戻るべきかについてパーサに示すためにある状態で使用される。
11	7:0	RW	parser_status	ある種の条件下で、パーサの状態を示すために使用される。
12	7:0	RO	parser_error_code	この位置はパーサが割り込みをしたときにエラーコードを含んでおり、処理を待っている。これが割り込みの原因を表す。
13	7	RW:0	parser_access	他のパーサレジスタへのアクセスを可能にするためには、このレジスタに値1が書き込まれなければならない。制御用マイクロプロセッサは次に、パーサがデータの処理を中止し、アクセス可能であることを示す値1を繰り返すまでこのビットをポーリングしなければならない。 特殊な場合として、もしパーサが割り込みが処理されるのを待つのを中止されたときには始めに parser_accessに1を書き込むことなしに parser_error_codeが読み出されることに注意すべきである。
	6:0	RW	reg_keyhole_addr	このレジスタはパーサの内部レジスタファイル内の、reg_keyhole_dataを介して書き込みが行われ、或いは読み出しがおこなわれる位置をアドレス指定するために用いられる。reg_keyhole_dataへの各アクセス（読み出し及び



229

230

7Fxx(16進)	ビット	Dir/Type	レジスタ	記述
				書き込み)はreg_keyhole_addrを1だけ増分することに注意すべきである。
14	7:0	RW	reg_keyhole_data	この位置からの読み出しはパーソナルレジスタファイルのreg_keyhole_addrによって示される位置から実際にデータを読み出す。同様にこの位置への書き込みはパーソナルレジスタファイルのreg_keyhole_addrによって示される位置に実際書き込みを行う。
15	7:0	(未使用)		
16	7:0	RW	user_keyhole_addr	このレジスタはuser_keyhole_dataを介して書き込み或いは読み出しが行われるユーザーデータRAM内の位置をアドレス指定するために使用される。 user_keyhole_dataへの各アクセス (読み出し或いは書き込み) はuser_keyhole_addrを1だけ増分させることに注意すべきである。
17	7:0	RW	user_keyhole_data	この位置からの読み出しはユーザーデータRAMのreg_keyhole_addrによって示される位置から実際にデータを読み出す。同様にこの位置への書き込みはユーザーデータRAMのreg_keyhole_addrによって示される位置に実際に書き込みを行う。

7Fxx(16進)	ビット	Dir/Type	レジスタ	記述
00	7:4	---	(パーソナルレジスタ)	
	3	RW a/o	parser_event	このビットはパーソナルエラー状態を検出した時にはいつでも設定される。もしparser_maskも1に設定されていれば、割り込みが生成される。 <sup>a</sup>
	2:0	---	(パーソナルレジスタ)	
01	7:4	---	(パーソナルレジスタ)	
	6	RW/o	parser_mask	上記 parser_event 参照。
	3:0	---	(パーソナルレジスタ)	

表100 パーソナルレジスタ

a イベントビットは単純なR/Wレジスタビットである。

b 全ての割り込みはchip\_maskが1に設定されることが、必要条件である。

エラーコード

パーソナルイベント状態を検出したときにはいつもparser\_eventをセットしている。もしparser\_maskが1に設定されていれば (ユーザーシステムがパーソナルイベントを処理することに関係していることを表している)、パーソナル処理を停止し、(chip\_maskが1に設定されていることを仮定して) 割り込み

40 が生成される。

【0592】割り込みに応答するのに際して制御用マイクロプロセッサはイベントの原因を決定するためにparser\_error\_codeを読み取らなければならない。表101はこれに関して定義されたエラーコードの完全なリストを提供するものである。制御用マイクロプロセッサが適切な方法でイベントに応答した後は、該プロセッサは本発明のパーソナル処理を再開することを許可しなければならない。これはparser\_eventに値1を書き込むことによってイベントをクリアすることによって行われる。

【0593】

\* \* 【表101】

コード	名 称	記 述
	ERR_USER_DATA	ユーザーデータに遭遇し、それはユーザーデータRAMに存在することを表す。

表101. パーサ・エラーコード

ユーザーデータの扱い

小量のユーザーデータはパーサから読み出すことができる。省略時には、全てのユーザーデータがスタートコード検出器によって廃棄される。これは、システムをその能力を超えるかもしれない大量のユーザーデータが不適切に使用されることから保護するためである。

【0594】ユーザーデータがパーサに届くことを許可するためには、レジスタdiscard\_userが0に設定されねばならない。ビットストリームにおいてユーザーデータに遭遇したときにはいつもデータのバイトはオン・チップのユーザーデータRAMにおいてバッファリングされる。このRAMは192バイトのデータがバッファリングされる空間を有している。ユーザーデータの全てのバイトが読まれたとき（あるいはRAMが一杯の時）パーサはイベント（ERR\_USER\_DATA）を生成し、これにより制御用マイクロプロセッサがRAMからデータを読み出すことが可能になる。

【0595】ユーザーデータRAMが読まれる前に、マイクロプロセッサは、まずparser\_accessを1に設定することによってパーサの内部レジスタにアクセスし、次にこのビットをそれが1を読み返すまでポーリングすることが必要である。ユーザーデータRAM内のバイト数はparser\_statusによって表示される。ユーザーデータRAMは直接的にアクセスすることはできない。その代わりに、user\_keyhole\_addrに読み込まれるアドレス（通常ゼロ）を書き込むことが必要であり、その後データはuser\_keyhole\_dataから読み出される。user\_keyhole\_dataから読み出しが行われる毎にuser\_keyhole\_addrが自動的にインクリメントされるため、ユーザーデータの適切な数のバイトを非常に迅速に読み出すことができる。

【0596】もしユーザーデータが192バイト以下しか無ければ、全てのデータは単一のイベントで扱われる。もし192バイト以上有れば、最初にERR\_USER\_DATAが生成された時にparser\_statusは192バイト含むことになる。イベントがクリアされた後（parser\_accessにゼロを書き込み次にparser\_eventに1を書き込むことにより）、マイクロコードは次に何をするかを決める為にparser\_continueに質問する。

【0597】もしparser\_continueが1であれば、パーサはユーザーデータの処理を続ける。ユーザーデータの残りのバイト（あるいは次の192バ

イト）がストリームから構文解析され、処理が繰り返される。しかしながら、もしparser\_continueが0であれば、パーサは残りのユーザーデータを廃棄し、通常のビデオ復号を進める。parser\_continueがゼロであっても、第1のERR\_USER\_DATAイベントが常に生成されることに注意すべきである。

ユーザーデータの量の制限

もしユーザーデータが使用されるべきであるときには、本発明によってビデオデータのリアルタイムの復号が保証されるために、これが制限されていることが重要である。制御用マイクロプロセッサの割り込み応答時間及びシステムからデータのバイトを読み出すのに必要とされる時間等の多くの外部の制約に依存するためユーザーデータの許容可能な限界を指定することは非常に難しい。ガイドとして、ユーザーデータの量は、およそ50μ（割り込み応答時間等を含む）の間にシステムから読み出されることが保証される量に制限されることが必要である。

ユーザーデータRAM

画面データの復号の間、使用者RAMは他の目的（例えば、隠し動きベクトルの格納）等のためのマイクロコード・プロセッサによって使用される。このことによって、RAM内にデータを放置すること及び後の使用のためにそのデータが保存されていると期待することは不可能である。

【0598】序文

本発明はビデオタイムスタンプの管理を補助する回路を含んでいる。MPEGシステムストリームパーサに関連する外部回路がクロック基準（プログラムされたクロック基準或いはシステムクロック基準等の適当なもの）を使用して安定した27MHzクロックを獲得しているものと仮定する。

【0599】従って、本発明による回路は、音声との同期を図るためにビデオの復号を適切な時間に開始するように配慮されており、その後、ビデオタイムスタンプを監視して継続した同期を確保している。エラーが無ければ、それ以降の補正は必要とされない。クロック基準情報をビデオデコーダに転送する必要があることが好ましい。ハードウェアは二つの領域に分割されており、それらはビデオタイムスタンプをロードするためのシステムの入力ステージに関連する回路と、ビデオパーサ回路に関連する実時間カウンタである。

システムの構成

本発明は 27MHz の SYSCLOCK から導出された規則的な間隔で増分されるカウンタを含む。タイムスタンプ管理のためのシステムはシステム外部で維持されているこのカウンタの第 2 のコピーに（概念上）依存している。これらの二つのカウンタは同一の信号でリセットされることにより同一の値に初期化される。その後、二つのカウンタは自走する。

【06000】本発明は“ビデオ時間”と称する内部の時間カウンタに対してそのタイムスタンプ管理を実行する。正確な比較が成されることを保証するために、ビデオタイムスタンプはシステムデコーダにより修正される。絶対時間を知る必要は無く、単に、画面がデコードされた実際の時間とデコードされるべきであった公称時間の間の差異が良い。

【06010】以下の式 1 はビデオ時間カウンタ及び修正\* RESET\_T-TIME ピンが最後に行使されている場合には、ビデオ時間カウン

タのコピーを持つよりは、単純に“時間”の値を記録する方が良いかも知れない。この情報及び“時間”の現在の値から、常にシステム内のビデオ時間の現在の内容を推断することが可能である。

【06020】修正タイムスタンプの適切な値を導出することが可能な適当な演算動作の再配置の使用が可能である。図 122 に示された如く、発明において用いられる修正タイムスタンプは 16 ビットのみを使用している。これは二つの方法で達成される。第 1 には、時間及びタイムスタンプ（修正タイムスタンプを導出するのに使用されたタイムスタンプ=式 2 参照）の間の差異は常に小であるから、より上位のビットを廃棄することができる。第 2 に、本発明はビデオを最も短いフレーム時間に表示するように制御するのみであるから、より下位のビットもまた必要とされず、4 ビットだけにシフトすることによって廃棄される。

【06030】こうして、時間情報の維持された 16 ビットは約 11.5 秒迄のタイミングエラーを約 180  $\mu$ s（フィールド時間の約 1%）の精度で扱うことが可能である。

【図面の簡単な説明】

【図 1】図 1 は本発明の好ましい実施例におけるデータの流れを示す図である。

【図 2】図 2 は 64  $\times$  32 RAM 内に 8 ビットデータのアドレス指定するために用いられる 13 ビットのワードの例を示す図である。

【図 3】図 3 は本発明において登録ファイルの機能的ブロック図である。

【図 4】図 4 は図 3 に示された登録ファイル内のデータの流れを示す図である。

【図 5】図 5 は本発明による登録ファイルアドレス復号化を示すブロック図である。

【図 6】図 6 は本発明によるマイクロコードラ状態マ

\* タイムスタンプの間の差異を実際の“時間”（クロック基準から導出された）とタイムスタンプととの間の差異に等しく設定することによってこれを表している。式 2 は単に修正タイムスタンプを導出するために変数を再構成したものである。

式 1: ビデオ時間-修正タイムスタンプ=タイムスタンプ-時間

式 2: 修正タイムスタンプ=ビデオ時間+（タイムスタンプ-時間）

図 122 は修正タイムスタンプを導出するための一つの可能な演算の構成を示している。実際は、実際の加算（及びシフト）は専用のハードウェアの中でよりむしろプロセッサ上で実行される可能性が大である。勿論、修正タイムスタンプの同一の数値を導出するために多くの他の方法が存在する。例えば、本発明の

シンのブロック図である。

【図 7】図 7 はアドレス指定用に用いられ、アドレスフィールド、置換フィールド及び置換ヘッダを有する本発明による固定幅ワードを示す図である。

【図 8】図 8 は本発明による演算コアの例を示すブロック図である。

【図 9】図 9 は本発明による入力データで IDCT を行う方法における基本ステップを示す図である。

【図 10】図 10 は本発明による IDCT システムの結合した簡単な 2 ステージのアーキテクチャを示すブロック図である。

【図 11】図 11 は図 10 に示した IDCT の主要システム成分を備えた集積回路の簡単なブロック図である。

【図 12】図 12 a ~ 図 12 b からなる図 12 は主要システム成分のうちの 1 つに相当する事前処理回路のブロック図である。

【図 13】図 13 a ~ 図 13 c からなる図 13 は好ましい実施例の IDCT システム内のタイミング及び制御信号間の関係を示すタイミング図である。

【図 14】図 14 a ~ 図 14 b からなる図 14 は IDCT システム内の共通処理回路を示すブロック図である。

【図 15】図 15 ~ 図 15 d からなる図 15 はシステムの他の主要成分に対応する事後処理回路を示すブロック図である。

【図 16】図 16 a ~ 図 16 b からなる図 16 は対のデータストリーム、置換 RAM 及び改善したバッファを有する IDCT を示す本発明に応じたブロック図である。

【図 17】図 17 a ~ 図 17 f からなる図 17 は図 16 に示した 1 次元 IDCT システムを詳細に示すブロック図である。

【図 18】図 18 a ~ 図 18 b からなる図 18 は図 17 に示した変換システムを更に大きく詳細に示すブロック図である。

【図19】図19a～図19bからなる図19は図18に示した入力バッファを更に大きく詳細に示すブロック図である。

【図20】図20a～図20bからなる図20は本発明による事前処理回路「PREC」を簡単に示すブロック図である。

【図21】図21a～図21bからなる図21はIDC T内の共通処理回路「CLK」を示すブロック図である。

【図22】図22a～図22bからなる図22は事後処理回路「POSTC」を示すブロック図である。

【図23】図23a～図23dからなる図23は図22に示した事後処理回路のその他を示す図である。

【図24】図24は本発明による四捨五入及び飽和ブロックを示すブロック図である。

【図25】図25a～図25bからなる図25は本発明における出力バッファを示すブロック図である。

【図26】図26a～図26bからなる図26は本発明における制御シフトレジスタを示すブロック図である。

【図27】図27a～図27cからなる図27は本発明における制御シフトレジスタ復号を示すブロック図である。

【図28】図28a～図28cからなる図28は制御シフトレジスタ及び入力制御バッファを示す図である。

【図29】図29a～図29fからなる図29はT2データストリーム用の制御回路を示す図である。

【図30】図30a～図30dからなる図30はT1用のカウンタ内のデータを示す図である。

【図31】図31a～図31eからなる図31は本発明においてT2データストリーム用のカウンタ内のデータを示す図である。

【図32】図32はIDC T及び関連回路を示すブロック図である。

【図33】図33はT1及びT2データのインターリーブを示すタイミング図である。

【図34】図34はT2のスリッページ及び復旧を示すタイミング図である。

【図35】図35は本発明においてIDC T及び関連回路のフラッシュ動作を示すタイミング図である。

【図36】図36は本発明によるシステムの始動を示す図である。

【図37】図37はT1及びT2データをインターリーブする早期ステージ内のスリッページ及び復旧を示すタイミング図である。

【図38】図38は図16～図37に示したIDC Tシステムの他の好ましい実施例を示す図である。

【図39】図39は本発明によるデータ及びタイムスタンプ情報を含む基本ストリームに多重分離されるMPEG情報ストリームを示す図である。

【図40】図40は本発明による基本ストリームタイム

スタンプエラー決定及び時間同期システムの第1の実施例を示す図である。

【図41】図41は本発明による基本ストリームタイムスタンプエラー決定及び時間同期システムの第2の実施例を示す図である。

【図42】図42は本発明による基本ストリームタイムスタンプエラー決定及び時間同期システムの第3の実施例を示す図である。

【図43】図43は本発明によるビデオタイムスタンプエラー決定及び時間同期システムの第1の実施例を示す図である。

【図44】図44は本発明によるビデオタイムスタンプエラー決定及び時間同期システムの第2の実施例を示す図である。

【図45】図45は図44に示した30Hzで動作するビデオタイムスタンプエラー決定及び時間同期システムの第2の実施例を示す図である。

【図46】図46は本発明のシステムを介して流れるタイムスタンプ情報を示す図である。

【図47】図47はマイクロプログラマブル状態マシンによって処理される同期時間情報を示すブロック図である。

【図48】図48は本発明の第1の好ましい実施例を示すブロック図である。

【図49】図49は本発明の第1の好ましい実施例を示す他のブロック図である。

【図50】図50は本発明の第2の好ましい実施例を示すブロック図である。

【図51】図51は本発明の第2の実施例によって用いられるアドレス指定方法を詳細に示す図である。

【図52】図52は本発明によるハフマンVLCを復号化する装置を示すブロック図である。

【図53】図53a～図53dからなる図53は本発明による並列ハフマンデコーダの構成を示す概略ブロック図である。

【図54】図54a～図54bからなる図54は並列ハフマン符号を復号化するために適合されたROMを示す概略ブロック図である。

【図55】図55は並列ハフマン符号を復号化するために適合されたROMの第1の実施例を示す図である。

【図56】図56は並列ハフマン符号を復号化するために適合されたROMの第2の実施例を示す図である。

【図57】図57は並列ハフマン符号を復号化するために適合されたROMの第3の実施例を示す図である。

【図58】図58は本発明の一実施例の主要システム成分を示すブロック図である。

【図59】図59は本発明のスタート符号検出器を示すブロック図である。

【図60】図60は本発明の分析器を示すブロック図である。

【図61】図61は本発明の空間処理回路の主要成分を示すブロック図である。

【図62】図62は本発明に応じた表示回路を示すブロック図である。

【図63】図63は本発明に応じたタイムスタンプ管理の一実施例を示す図である。

【図64】図64は本発明におけるタイムスタンプ管理の他の実施例を示す図である。

【図65】図65は本発明のシステムのハードウェア成分を示すブロック図である。

【図66】図66は本発明のマイクロコントローラのシステム成分の概略を示すブロック図である。

【図67】図67は本発明の演算コアを示す簡単な図である。

【図68】図68は本発明のALUを示す図である。

【図69】図69は本発明に応じた登録ファイルを示す図である。

【図70】図70は本発明において独立のバスレジスタへの書き込みを示す図である。

【図71】図71はベクトル[1]=0及びベクトル[0]=0におけるフレームに基づいた予測を示す図である。

【図72】図72はベクトル[1]=0及びベクトル[0]=1におけるフレームに基づいた予測を示す図である。

【図73】図73はベクトル[1]=1及びベクトル[0]=0におけるフレームに基づいた予測を示す図である。

【図74】図74はベクトル[1]=1及びベクトル[0]=1におけるフレームに基づいた予測を示す図である。

【図75】図75はmotion\_vertical\_field\_select=0及びベクトル[0]=0におけるフィールドに基づいた予測を示す図である。

【図76】図76はmotion\_vertical\_field\_select=0及びベクトル[0]=1におけるフィールドに基づいた予測を示す図である。

【図77】図77はmotion\_vertical\_field\_select=1及びベクトル[0]=0におけるフィールドに基づいた予測を簡単に示す図である。

【図78】図78はmotion\_vertical\_field\_select=1及びベクトル[0]=1におけるフィールドに基づいた予測を示す図である。

【図79】図79はmotion\_vertical\_field\_select=0及びベクトル[0]=0におけるフレーム映像のフィールドに基づいた予測を示す図である。

【図80】図80はmotion\_vertical\_field\_select=0及びベクトル[0]=1

10

20

30

40

50

におけるフレーム映像のフィールドに基づいた予測を示す図である。

【図81】図81はmotion\_vertical\_field\_select=1及びベクトル[0]=0におけるフレーム映像のフィールドに基づいた予測を示す図である。

【図82】図82はmotion\_vertical\_field\_select=1及びベクトル[0]=1におけるフレーム映像のフィールドに基づいた予測を示す図である。

【図83】図83は予測フィルタリングの追加モードを示す図である。

【図84】図84は他の予測モードを示す図である。

【図85】図85は本発明による他の予測モードを示す図である。

【図86】図86は本発明による他の予測モードを示す図である。

【図87】図87は本発明の表示システムの様々なシステム成分の構成を示すブロック図である。

【図88】図88は4:3フィルタリング動作を示す図である。

【図89】図89は3:2フィルタリング動作を示す図である。

【図90】図90は本発明の2:1フィルタリング動作を示す図である。

【図91】図91は本発明において用いられる3つのタップフィルタを示す図である。

【図92】図92は間違っただ画像の反復を示す図である。

【図93】図93は本発明のfield\_id信号を示す図である。

【図94】図94は本発明に応じた水平タイミング点(サイクル)を示す図である。

【図95】図95は本発明に応じたフィールド当たり62.5ラインでPAL垂直タイミングを示す図である。

【図96】図96は本発明に応じたフィールド当たり52.5ラインでNTSC垂直タイミングを示す図である。

【図97】図97は本発明に応じた水平計数機器を示す図である。

【図98】図98は本発明における境界発生を示す図である。

【図99】図99は本発明に応じた映像切り落としを示す図である。

【図100】図100はチップとして本発明を示すブロック図である。

【図101】図101は本発明のシステムクロック必要条件を示す図である。

【図102】図102は本発明に応じた符号化データインターフェースの2線プロトコルを示す図である。

【図103】図103は本発明のDATAトークンを示す図である。

す図である。

【図104】図104は本発明のFLUSHトークンを示す図である。

【図105】図105は符号化データインターフェースのタイミングを示す図である。

【図106】図106は本発明に応じた均等でないマーカースペース空間比CDCLOCKの使用を示す図である。

【図107】図107は本発明における16ビットモードにおける出力タイミングを示す図である。

【図108】図108は本発明における8ビットモードにおける出力タイミングを示す図である。

【図109】図109は本発明におけるビデオ出力インターフェースのタイミングを示す図である。

【図110】図110は本発明に応じたビデオ出力モード信号を示す図である。

【図111】図111は本発明における水平タイミングを示す図である。

【図112】図112 a～図112 bからなる図112は525ラインシステム用の垂直タイミングを示す図である。

【図113】図113 a～図113 bからなる図113は625ラインシステム用の垂直タイミングを示す図である。

【図2】

アドレス指定用固定幅ワード					
幅指定フィールド		アドレスフィールド			置換表示
継続マーカ	終端マーカ		置換フィールド	継続マーカ	
			置換フィールド	継続マーカ	
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
000	1	1101	1	000	0
111	0	1101	0	111	1

図2

【図7】

アドレス指定用固定幅ワード					
幅指定フィールド		アドレスフィールド			置換表示
継続マーカ	終端マーカ		置換フィールド	継続マーカ	
			置換フィールド	継続マーカ	
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000

図7

\*ある。

【図114】図114は本発明に応じた525ラインシステム用の同期及びブランキング信号を示す図である。

【図115】図115は本発明に応じた625ラインシステム用の同期及びブランキング信号を示す図である。

【図116】図116は本発明における0のSDRAM接続構成を示す図である。

【図117】図117は本発明における1のSDRAM接続構成を示す図である。

【図118】図118は本発明における2のSDRAM接続構成を示す図である。

【図119】図119は本発明における3のSDRAM接続構成を示す図である。

【図120】図120は本発明に応じたflag\_picture\_end動作を示すフローチャートである。

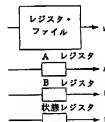
【図121】図121は本発明に応じたstart\_code\_search動作を示すフローチャートである。

【図122】図122は本発明に応じたタイムスタンプ変更を示す図である。

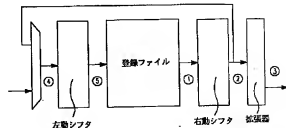
【図123】図123はマイクロプロセッサインターフェース用の読出タイミングを示す図である。

【図124】図124はマイクロプロセッサインターフェース用の書込タイミングを示す図である。

【図69】

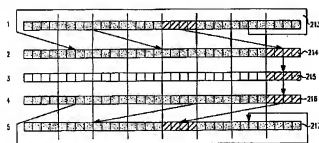


【図4】

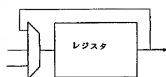




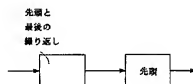
【図3】



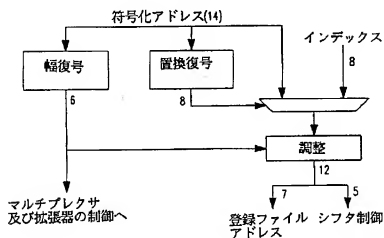
【図70】



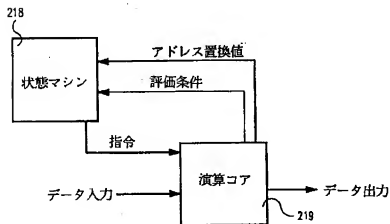
【図91】



【図5】

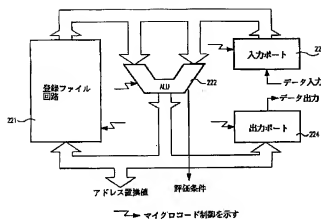


【図6】

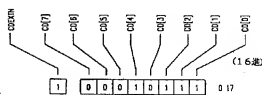




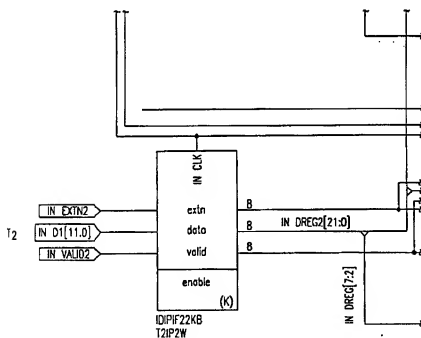
【図8】



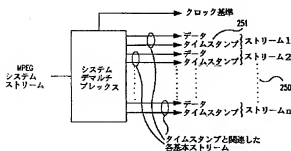
【図104】



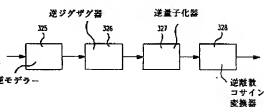
【図17b】



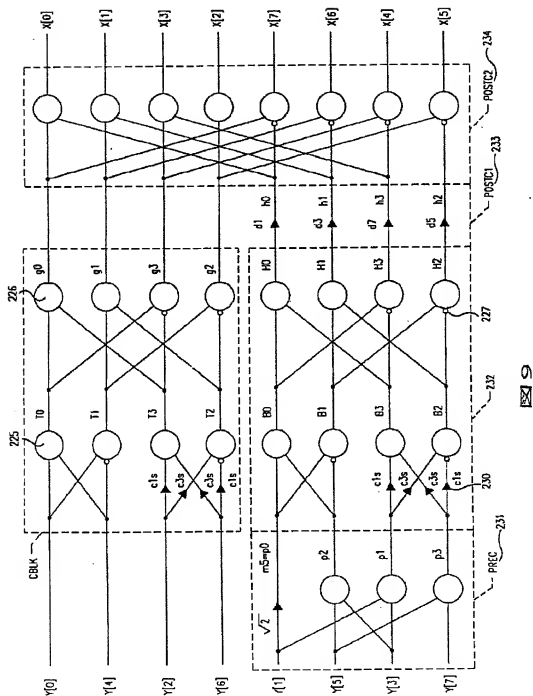
【図39】



【図61】



【図9】



【図 10】

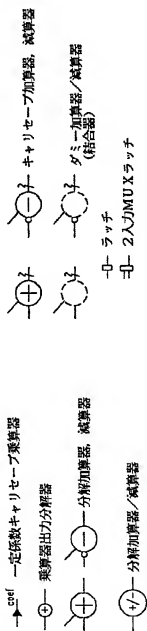
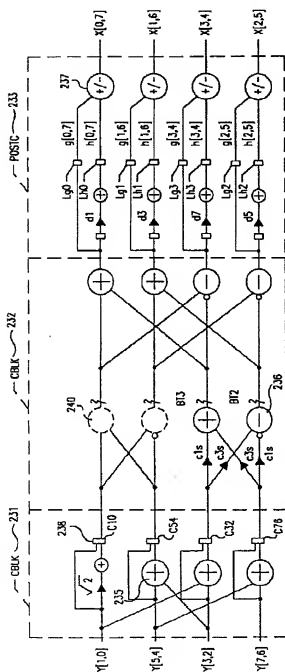
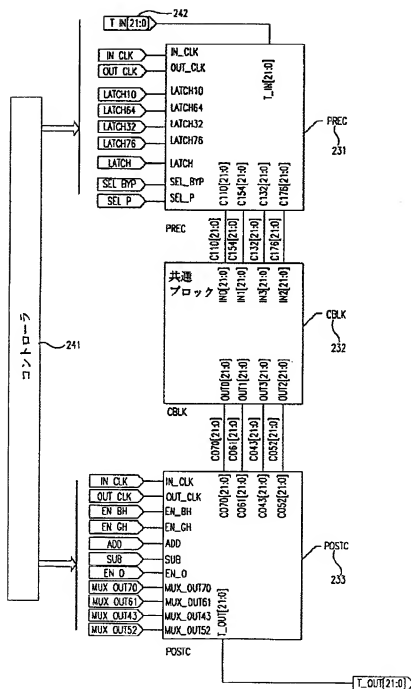


図 10

【図 11】



【図12a】

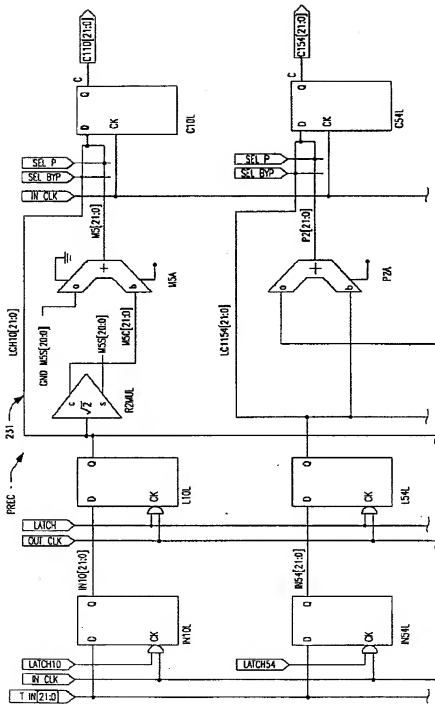
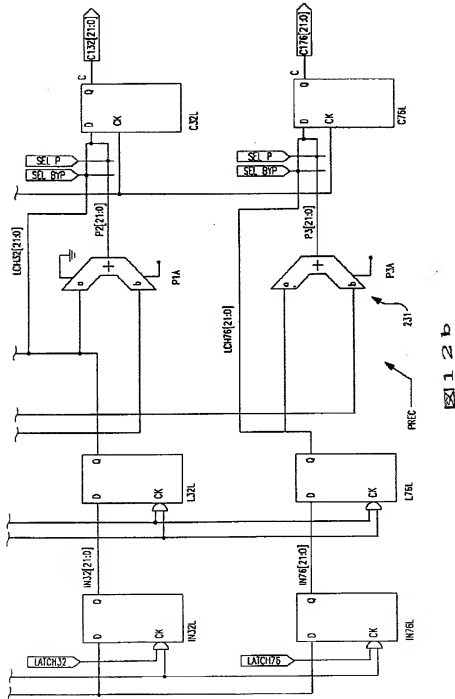


図 12 a

【図12b】



【図13a】

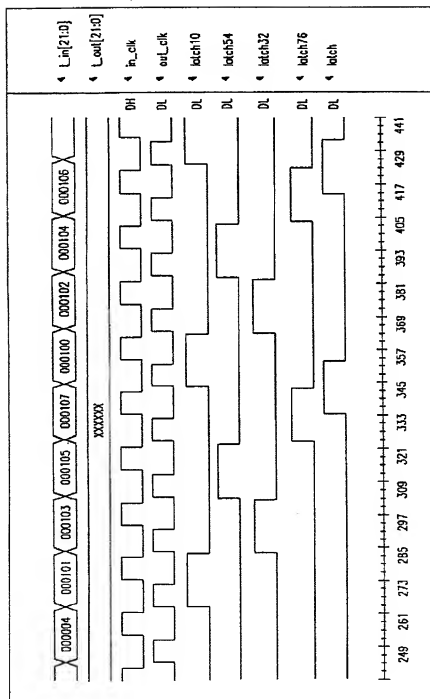


図13a

【図13b】

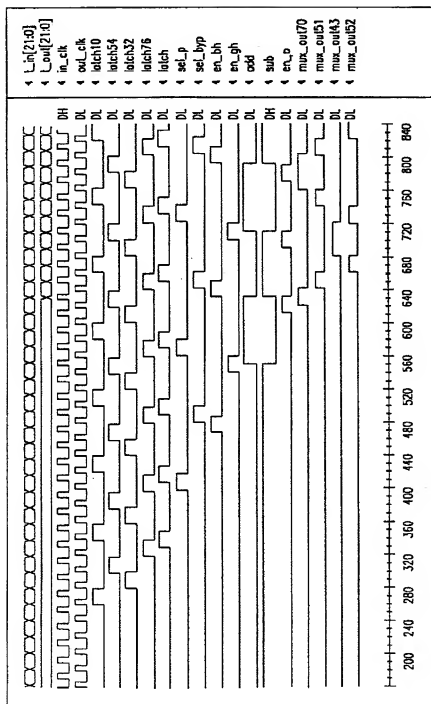
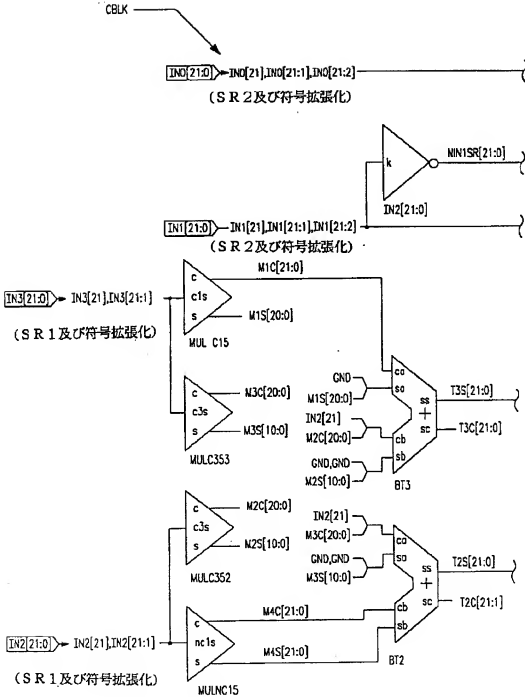


図13b

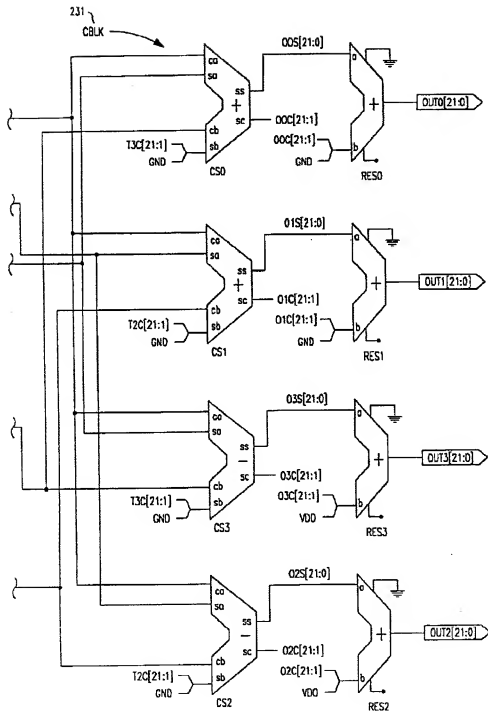




【図14a】



【図14b】



【図15a】

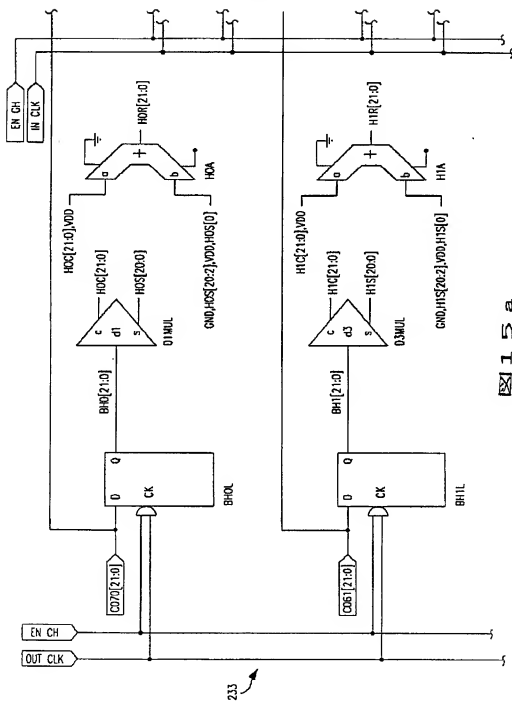


図15a

【図15b】

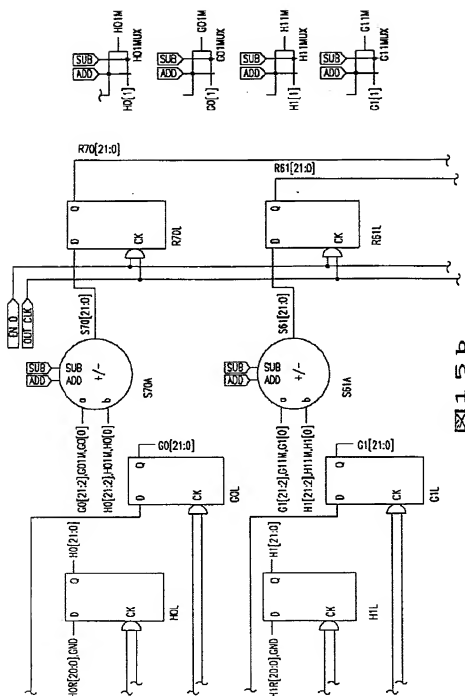


図 15 b

【図15c】

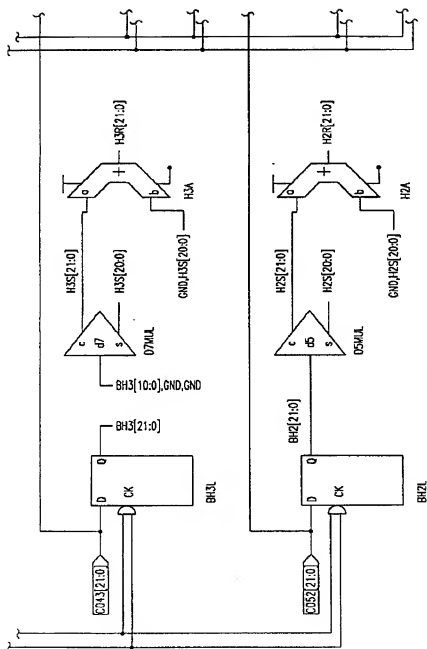
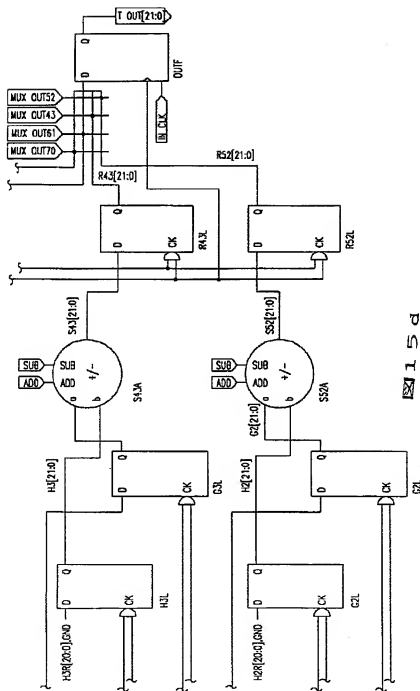


図 15 c

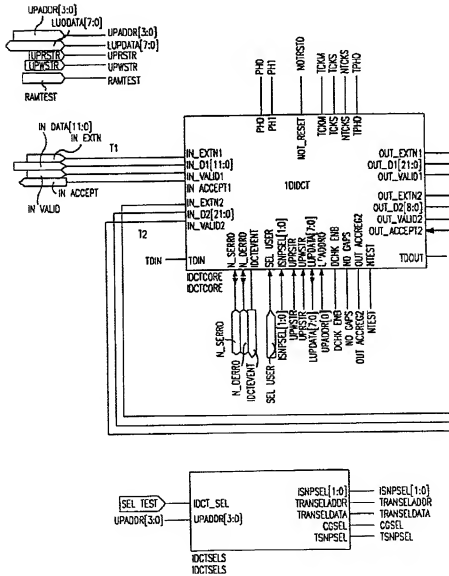
**15d**



Pin diagram of the FM78C01B chip. The chip is rectangular with pins numbered 1 to 20. The pins are labeled as follows:

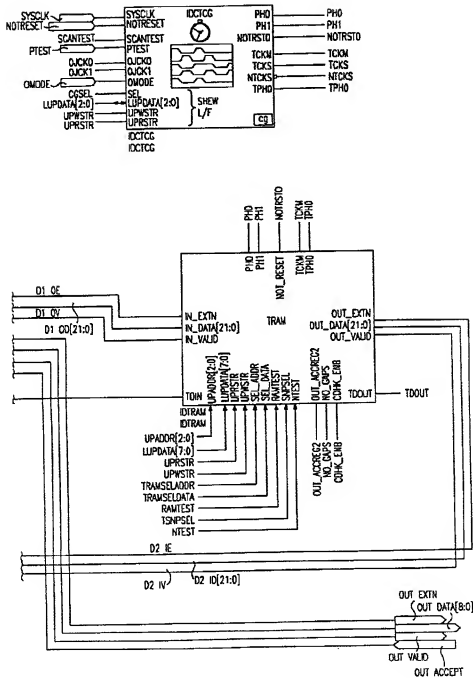
- Pin 1: TDOUT
- Pin 2: FM\_SCAN
- Pin 3: FM\_SCAN\_INVERT
- Pin 4: SCAN
- Pin 5: TO\_SCAN
- Pin 6: SCANTEST
- Pin 7: TMIN
- Pin 8: SCANTEST
- Pin 9: NTADDR\_OUT
- Pin 10: NADDR\_OUT
- Pin 11: NTADDR\_OUT
- Pin 12: NTADDR\_OUT
- Pin 13: NTADDR\_OUT
- Pin 14: NTADDR\_OUT
- Pin 15: NTADDR\_OUT
- Pin 16: NTADDR\_OUT
- Pin 17: NTADDR\_OUT
- Pin 18: NTADDR\_OUT
- Pin 19: NTADDR\_OUT
- Pin 20: NTADDR\_OUT

The chip is labeled "FM78C01B" and "JANUARY 1984".



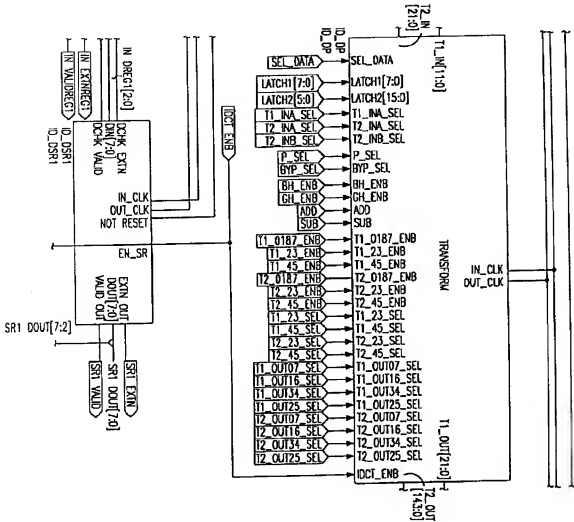


【図16b】

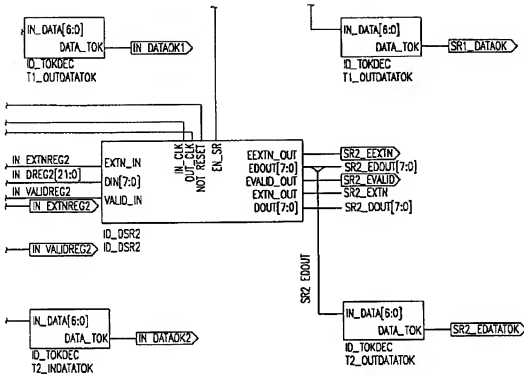




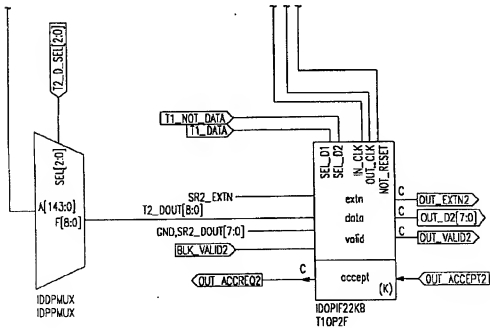
【図17c】



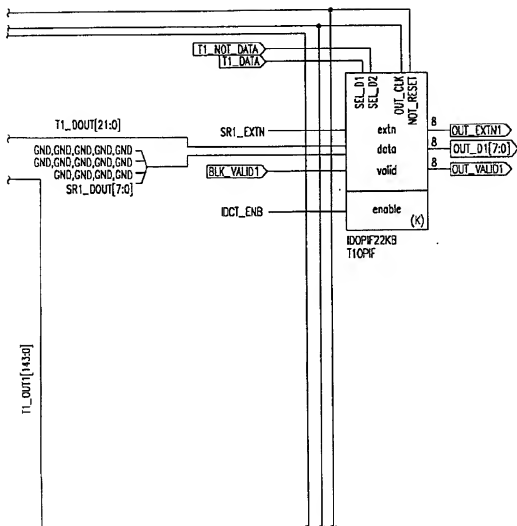
【図 17 d】



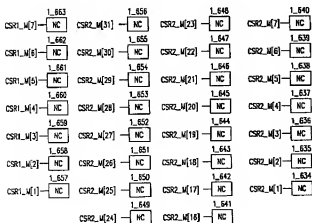
【図 17 f】



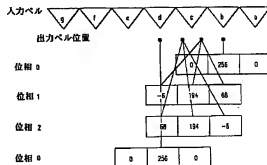
【図 17 e】



【図 27 b】



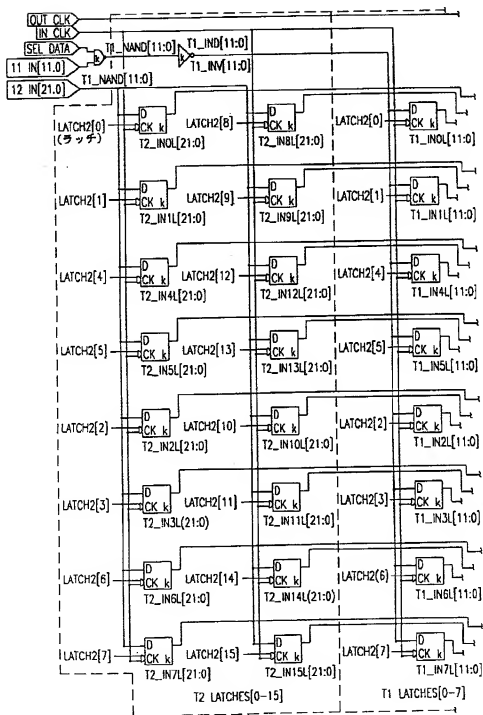
【図 89】



[illegible]

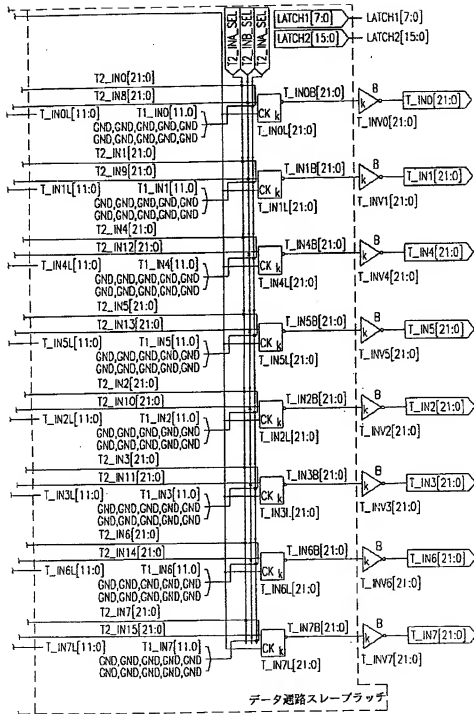


【図19a】

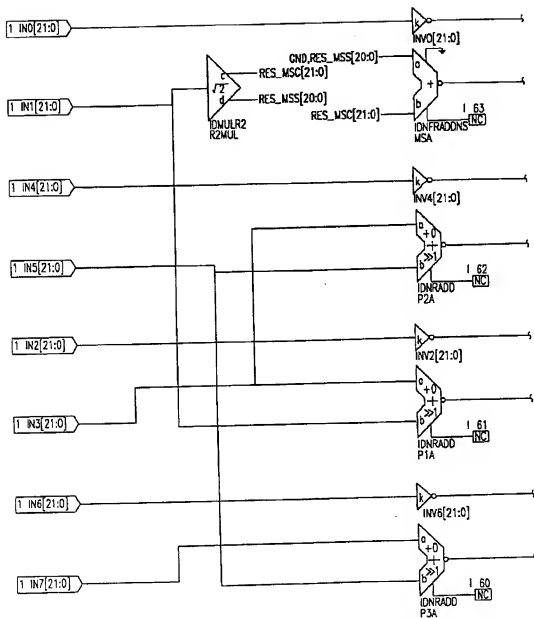




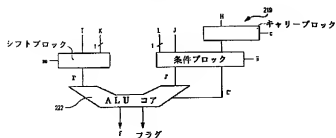
【図 19 b】



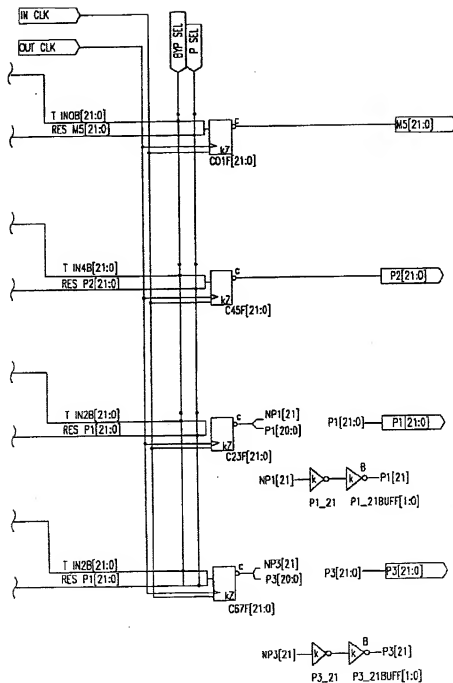
【図 20 a】



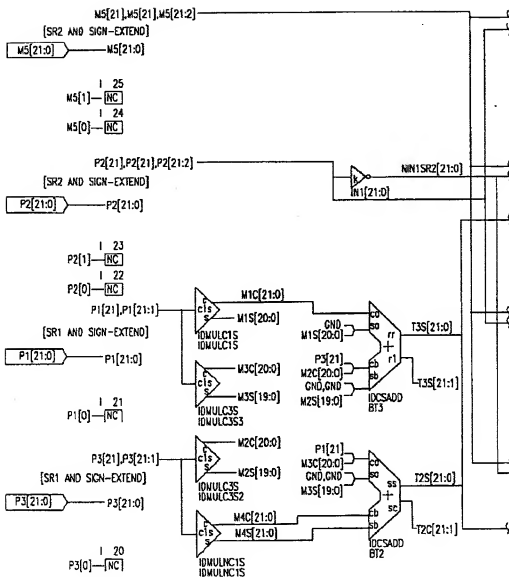
【図 6 8】



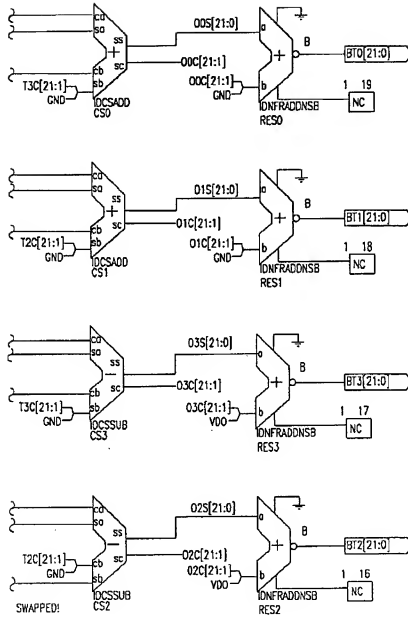
【図 20 b】



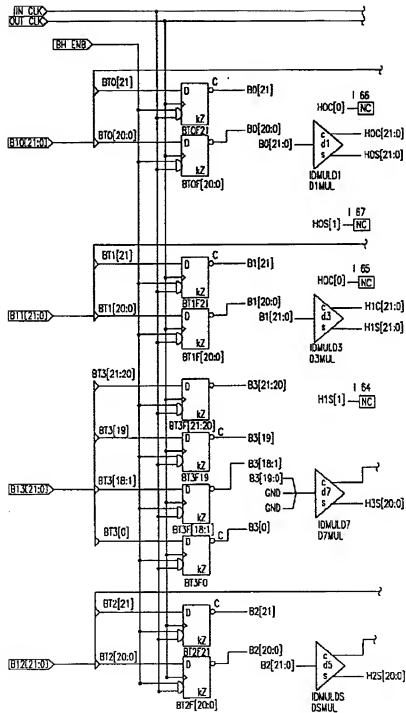
【図 21 a】



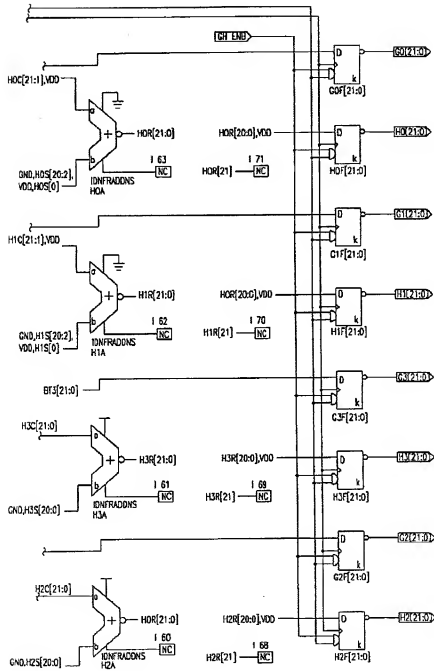
【図 21 b】



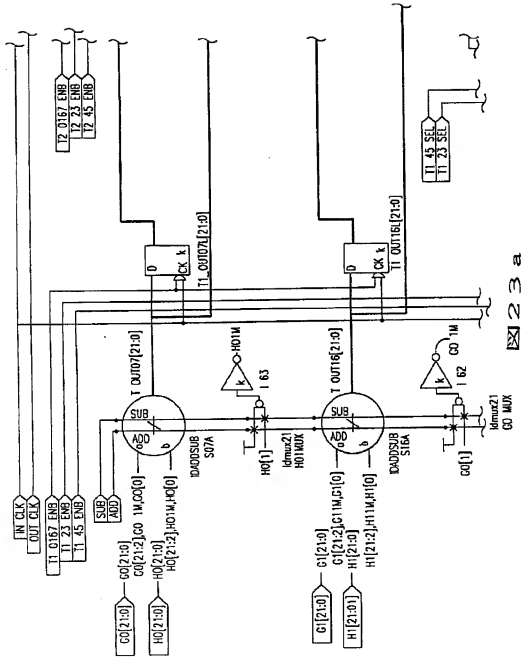
【図 22 a】



【図 22 b】



【図23a】





【図 23 b】

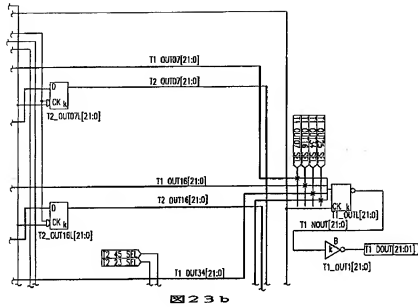


図 23 b

【図 23 c】

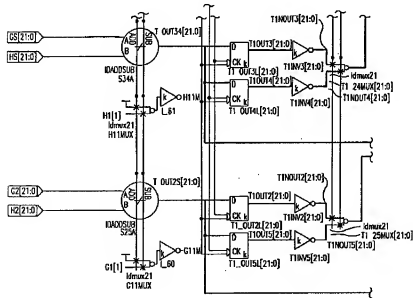
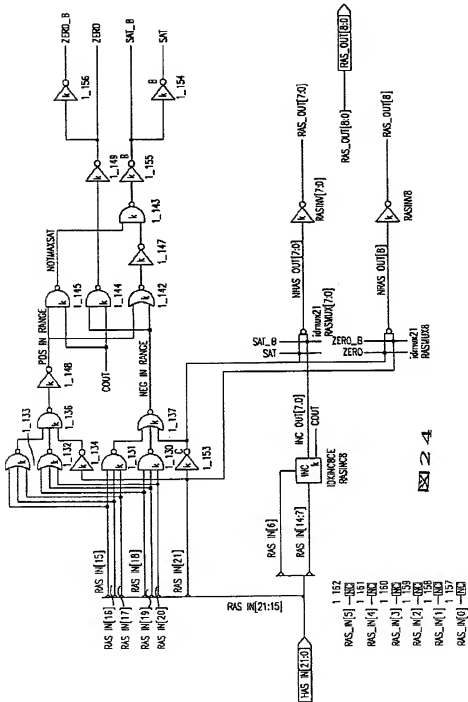


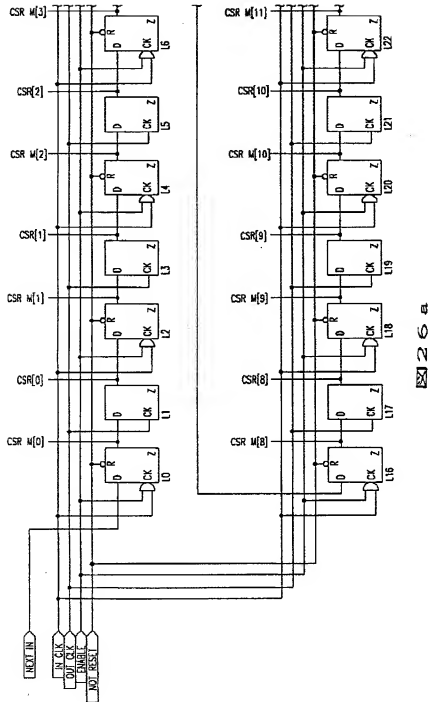
図 23 c



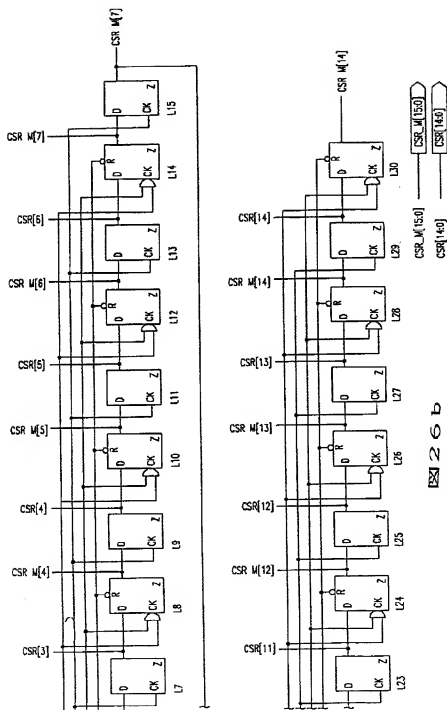
【图 2 4】



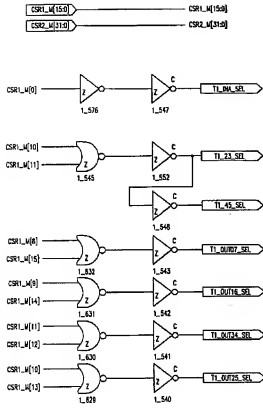
【図 2 6 a】



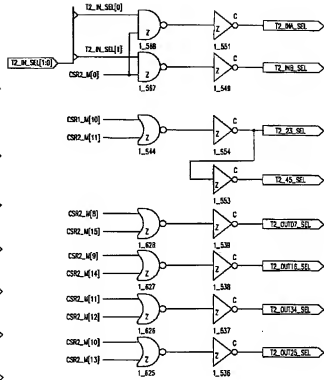
【図 26 b】



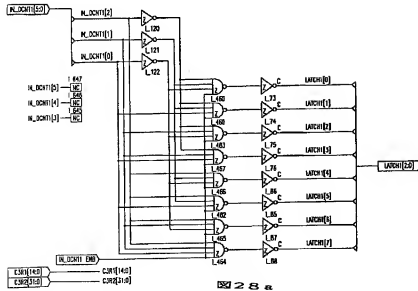
【図 27 a】



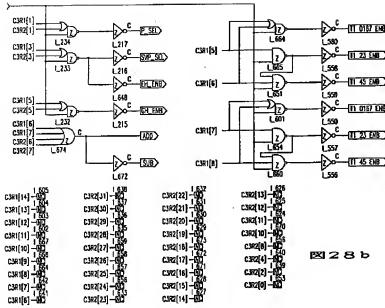
【図 27 c】



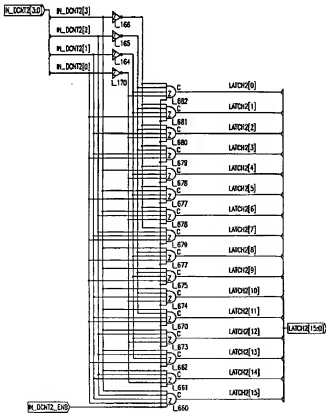
【図 28 a】



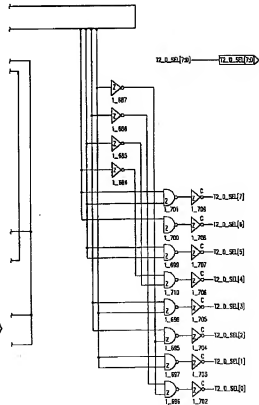
【图 28 b】



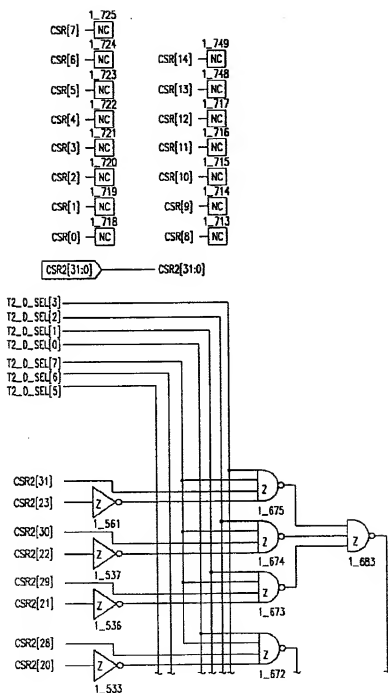
【図28c】



【図 29 d】

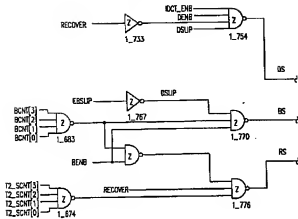


【図 29 a-1】

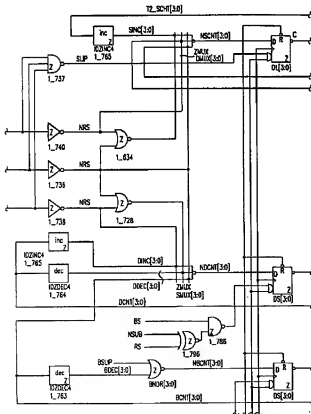




【図 29 b】



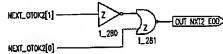
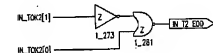
【図 29 c】



【図 31 c】

標準データトークンデコードはデータ、なし及び  
その他のトークンを認識する  
符号化>

token[1]	token[0]	
0	0	なし
0	0	その他
1	1	データ
1	1	不適





【図 30 b】

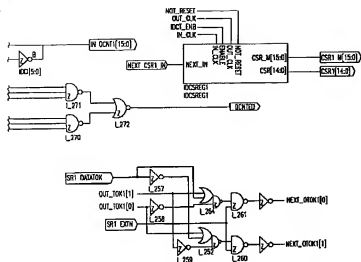


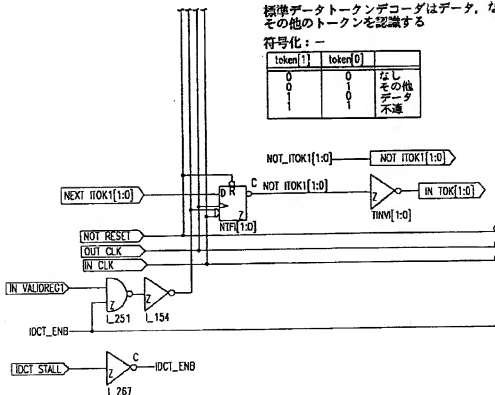
図 30 b

【図 30 c】

標準データトークンデコーダはデータ、なし及び  
その他のトークンを認識する

符号化: -

token[1]	token[0]	
0	0	なし
0	1	その他
1	0	データ
1	1	不通



【図 30 d】

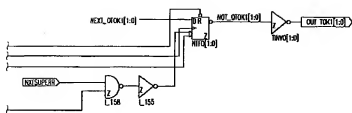


図 30 d

【図 31 a】

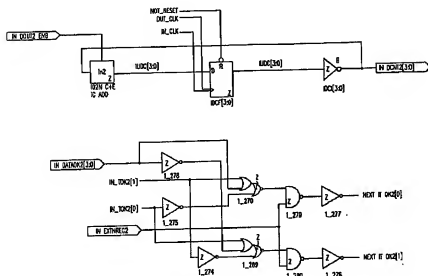


図 31 a

【図 59】

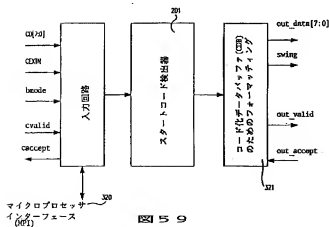


図 59

【図 31 b】

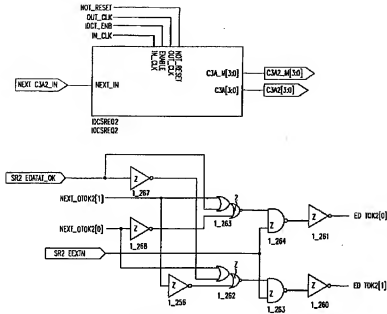


図 31 b

【図 31 d】

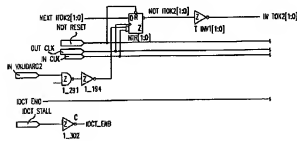
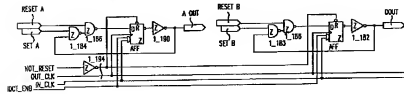
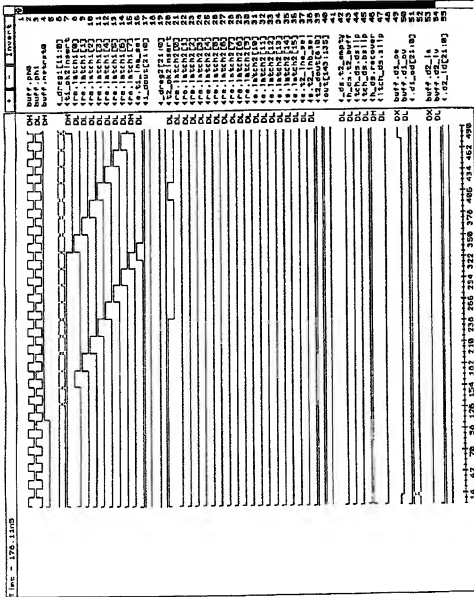


図 31 d



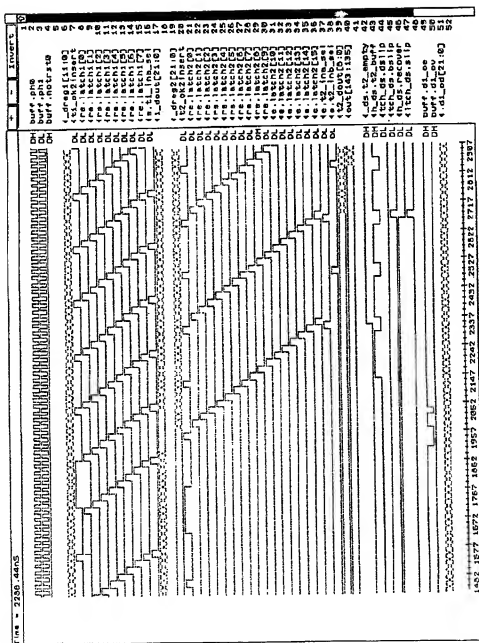


【图 3 2】



232

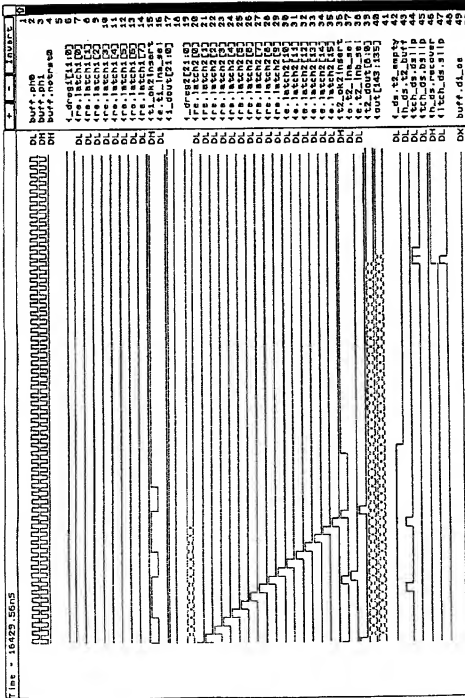
【图 3 3】







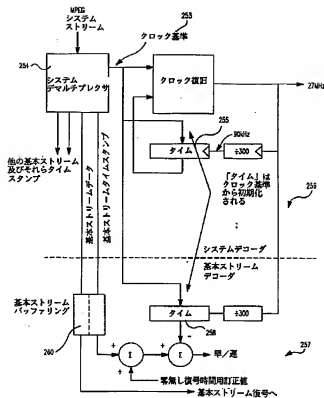
【図 35】



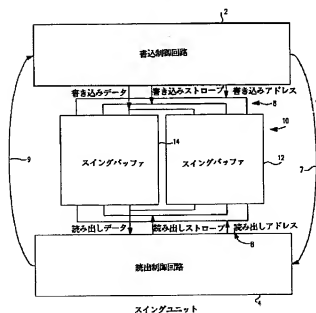




【図 40】



【図 48】



The diagram illustrates the process of generating a basic stream code from an MPEG system stream. Key components and their interactions are as follows:

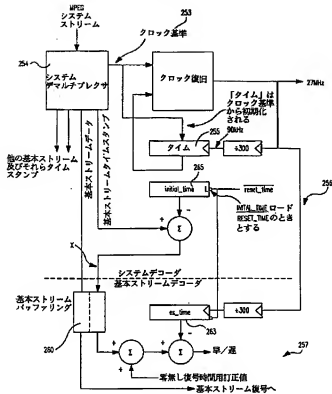
- MPEGシステムストリーム (MPEG System Stream):** The input stream entering the system.
- システムデマルチプレクサ (System Demultiplexer):** Receives the MPEG stream and outputs multiple streams, including the basic stream and other streams with their own timing stamps.
- クロック基準 (Clock Reference):** A reference clock signal used for synchronization.
- クロック復旧 (Clock Recovery):** A block that recovers the clock signal from the stream, outputting a 27MHz signal.
- タイム (Time):** A block that calculates the time difference between the recovered clock and the reference clock, outputting a 255 kHz signal.
- es\_line (Elementary Stream Line):** A signal that indicates the position of the elementary stream within the system stream.
- 基本ストリームデコード (Basic Stream Decoding):** A block that decodes the basic stream, outputting a 260 MHz signal.
- 基本ストリームパッパリング (Basic Stream Padding):** A block that pads the basic stream to a fixed length.
- 無誤し復号時時間訂正 (Error-free Decoding Time Correction):** A block that corrects the time of the decoded stream.
- 基本ストリーム復号へ (To Basic Stream Decoding):** The final output of the system, which is fed back into the clock recovery block.

The diagram also includes several numerical values and labels that provide context for the timing and processing steps:

- 254:** Label for the System Demultiplexer.
- 253:** Label for the Clock Reference input.
- 270kHz:** Output of the Clock Recovery block.
- 255 kHz:** Output of the Time block.
- 256:** Label for the Time block.
- 257:** Label for the es\_line signal.
- 258:** Label for the es\_line signal.
- 259:** Label for the es\_line signal.
- 260 MHz:** Output of the Basic Stream Decoding block.
- 261:** Label for the es\_line signal.
- 262:** Label for the es\_line signal.
- 263:** Label for the es\_line signal.
- 264:** Label for the es\_line signal.
- 265:** Label for the es\_line signal.
- 266:** Label for the es\_line signal.
- 267:** Label for the es\_line signal.
- 268:** Label for the es\_line signal.
- 269:** Label for the es\_line signal.
- 270:** Label for the es\_line signal.
- 271:** Label for the es\_line signal.
- 272:** Label for the es\_line signal.
- 273:** Label for the es\_line signal.
- 274:** Label for the es\_line signal.
- 275:** Label for the es\_line signal.
- 276:** Label for the es\_line signal.
- 277:** Label for the es\_line signal.
- 278:** Label for the es\_line signal.
- 279:** Label for the es\_line signal.
- 280:** Label for the es\_line signal.
- 281:** Label for the es\_line signal.
- 282:** Label for the es\_line signal.
- 283:** Label for the es\_line signal.
- 284:** Label for the es\_line signal.
- 285:** Label for the es\_line signal.
- 286:** Label for the es\_line signal.
- 287:** Label for the es\_line signal.
- 288:** Label for the es\_line signal.
- 289:** Label for the es\_line signal.
- 290:** Label for the es\_line signal.
- 291:** Label for the es\_line signal.
- 292:** Label for the es\_line signal.
- 293:** Label for the es\_line signal.
- 294:** Label for the es\_line signal.
- 295:** Label for the es\_line signal.
- 296:** Label for the es\_line signal.
- 297:** Label for the es\_line signal.
- 298:** Label for the es\_line signal.
- 299:** Label for the es\_line signal.
- 300:** Label for the es\_line signal.
- 301:** Label for the es\_line signal.
- 302:** Label for the es\_line signal.
- 303:** Label for the es\_line signal.
- 304:** Label for the es\_line signal.
- 305:** Label for the es\_line signal.
- 306:** Label for the es\_line signal.
- 307:** Label for the es\_line signal.
- 308:** Label for the es\_line signal.
- 309:** Label for the es\_line signal.
- 310:** Label for the es\_line signal.
- 311:** Label for the es\_line signal.
- 312:** Label for the es\_line signal.
- 313:** Label for the es\_line signal.
- 314:** Label for the es\_line signal.
- 315:** Label for the es\_line signal.
- 316:** Label for the es\_line signal.
- 317:** Label for the es\_line signal.
- 318:** Label for the es\_line signal.
- 319:** Label for the es\_line signal.
- 320:** Label for the es\_line signal.
- 321:** Label for the es\_line signal.
- 322:** Label for the es\_line signal.
- 323:** Label for the es\_line signal.
- 324:** Label for the es\_line signal.
- 325:** Label for the es\_line signal.
- 326:** Label for the es\_line signal.
- 327:** Label for the es\_line signal.
- 328:** Label for the es\_line signal.
- 329:** Label for the es\_line signal.
- 330:** Label for the es\_line signal.
- 331:** Label for the es\_line signal.
- 332:** Label for the es\_line signal.
- 333:** Label for the es\_line signal.
- 334:** Label for the es\_line signal.
- 335:** Label for the es\_line signal.
- 336:** Label for the es\_line signal.
- 337:** Label for the es\_line signal.
- 338:** Label for the es\_line signal.
- 339:** Label for the es\_line signal.
- 340:** Label for the es\_line signal.
- 341:** Label for the es\_line signal.
- 342:** Label for the es\_line signal.
- 343:** Label for the es\_line signal.
- 344:** Label for the es\_line signal.
- 345:** Label for the es\_line signal.
- 346:** Label for the es\_line signal.
- 347:** Label for the es\_line signal.
- 348:** Label for the es\_line signal.
- 349:** Label for the es\_line signal.
- 350:** Label for the es\_line signal.
- 351:** Label for the es\_line signal.
- 352:** Label for the es\_line signal.
- 353:** Label for the es\_line signal.
- 354:** Label for the es\_line signal.
- 355:** Label for the es\_line signal.
- 356:** Label for the es\_line signal.
- 357:** Label for the es\_line signal.
- 358:** Label for the es\_line signal.
- 359:** Label for the es\_line signal.
- 360:** Label for the es\_line signal.
- 361:** Label for the es\_line signal.
- 362:** Label for the es\_line signal.
- 363:** Label for the es\_line signal.
- 364:** Label for the es\_line signal.
- 365:** Label for the es\_line signal.
- 366:** Label for the es\_line signal.
- 367:** Label for the es\_line signal.
- 368:** Label for the es\_line signal.
- 369:** Label for the es\_line signal.
- 370:** Label for the es\_line signal.
- 371:** Label for the es\_line signal.
- 372:** Label for the es\_line signal.
- 373:** Label for the es\_line signal.
- 374:** Label for the es\_line signal.
- 375:** Label for the es\_line signal.
- 376:** Label for the es\_line signal.
- 377:** Label for the es\_line signal.
- 378:** Label for the es\_line signal.
- 379:** Label for the es\_line signal.
- 380:** Label for the es\_line signal.
- 381:** Label for the es\_line signal.
- 382:** Label for the es\_line signal.
- 383:** Label for the es\_line signal.
- 384:** Label for the es\_line signal.
- 385:** Label for the es\_line signal.
- 386:** Label for the es\_line signal.
- 387:** Label for the es\_line signal.
- 388:** Label for the es\_line signal.
- 389:** Label for the es\_line signal.
- 390:** Label for the es\_line signal.
- 391:** Label for the es\_line signal.
- 392:** Label for the es\_line signal.
- 393:** Label for the es\_line signal.
- 394:** Label for the es\_line signal.
- 395:** Label for the es\_line signal.
- 396:** Label for the es\_line signal.
- 397:** Label for the es\_line signal.
- 398:** Label for the es\_line signal.
- 399:** Label for the es\_line signal.
- 400:** Label for the es\_line signal.
- 401:** Label for the es\_line signal.
- 402:** Label for the es\_line signal.
- 403:** Label for the es\_line signal.
- 404:** Label for the es\_line signal.
- 405:** Label for the es\_line signal.
- 406:** Label for the es\_line signal.
- 407:** Label for the es\_line signal.
- 408:** Label for the es\_line signal.
- 409:** Label for the es\_line signal.
- 410:** Label for the es\_line signal.
- 411:** Label for the es\_line signal.
- 412:** Label for the es\_line signal.
- 413:** Label for the es\_line signal.
- 414:** Label for the es\_line signal.
- 415:** Label for the es\_line signal.
- 416:** Label for the es\_line signal.
- 417:** Label for the es\_line signal.
- 418:** Label for the es\_line signal.
- 419:** Label for the es\_line signal.
- 420:** Label for the es\_line signal.
- 421:** Label for the es\_line signal.
- 422:** Label for the es\_line signal.
- 423:** Label for the es\_line signal.
- 424:** Label for the es\_line signal.
- 425:** Label for the es\_line signal.
- 426:** Label for the es\_line signal.
- 427:** Label for the es\_line signal.
- 428:** Label for the es\_line signal.
- 429:** Label for the es\_line signal.
- 430:** Label for the es\_line signal.
- 431:** Label for the es\_line signal.
- 432:</**

Figure 4 is a block diagram of a dual-channel system. It consists of two main processing units, one on the left and one on the right. Each unit contains a RAM array (RAMアレイ) and a read circuit (読み出し回路). The left unit has a data input (データ入力) at 30 and a data output (データ出力) at 22. The right unit has a write address input (書き込みアドレス) at 10 and a data output (データ出力) at 22. Both units are connected to a common bus system, with labels 12, 14, 16, 18, and 20 indicating various internal connections and outputs. The right unit also has a read address output (読み出しアドレス) at 24.

【图 4 2】



【图 5 1】

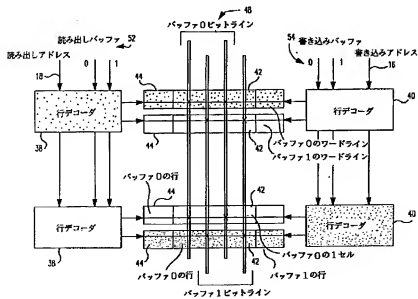


图 5-1

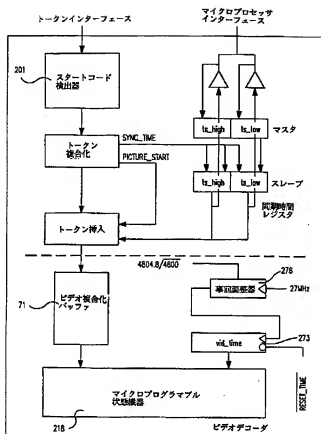




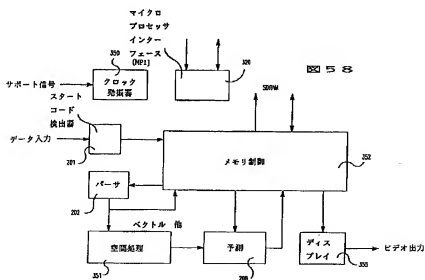




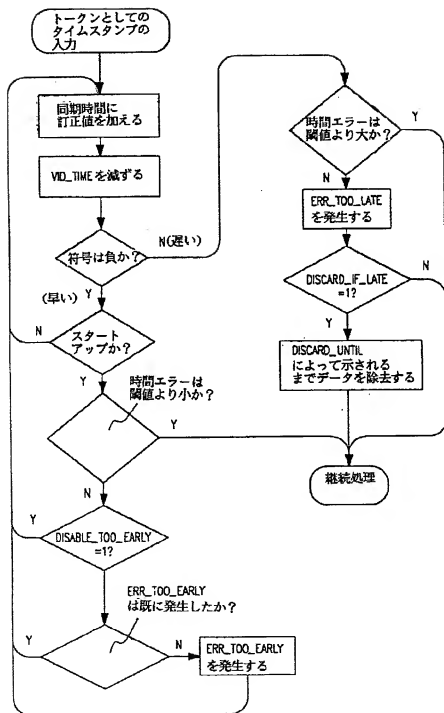
【図46】



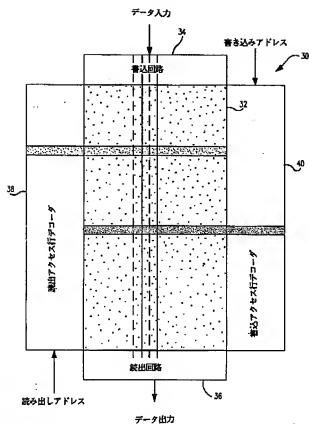
【図58】



【図47】



【図50】



【図60】

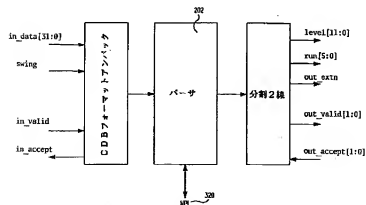
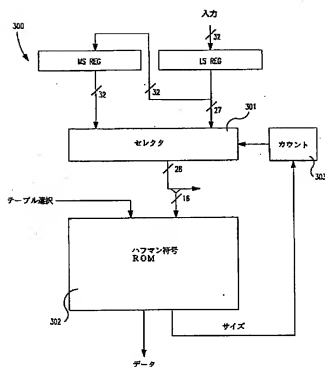
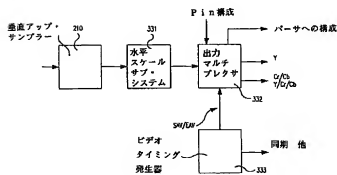


図 60

【図52】

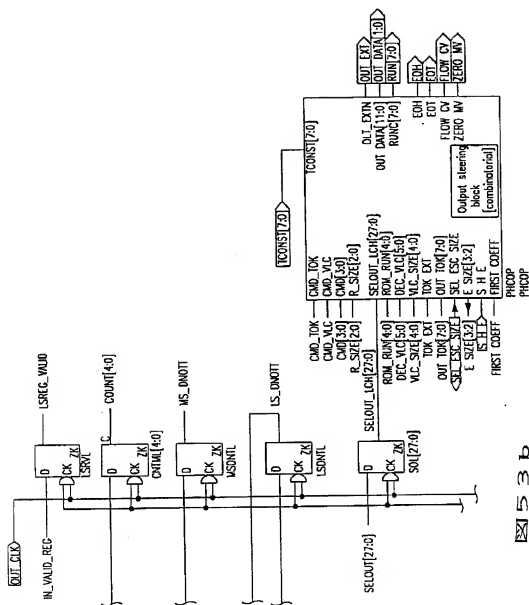


【図62】



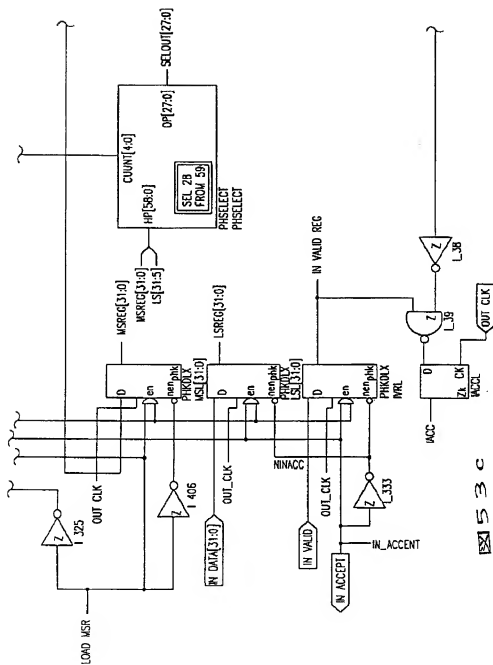


【図 53 b】

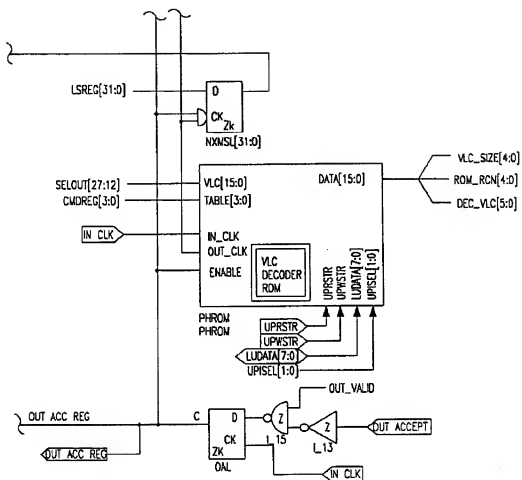




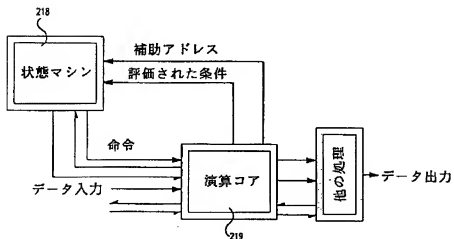
【図 53 c】



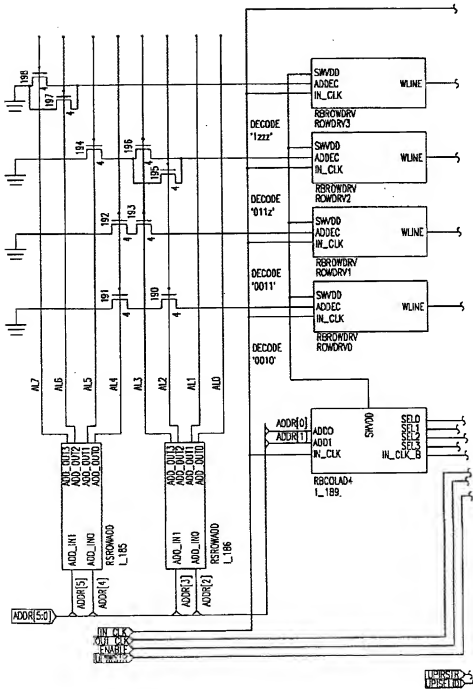
【図 53 d】



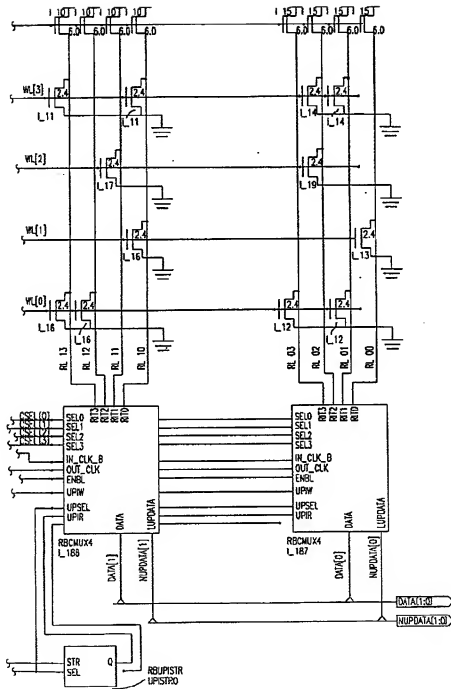
【図 66】



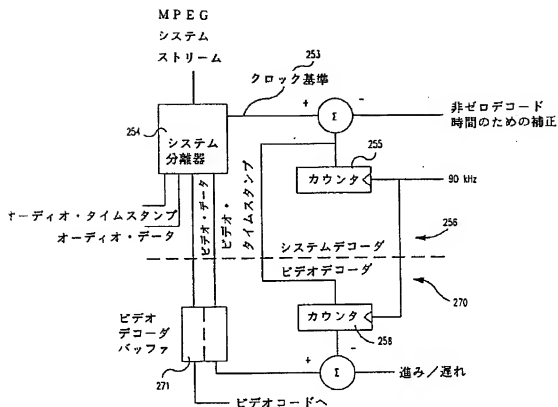
【図 54 a】



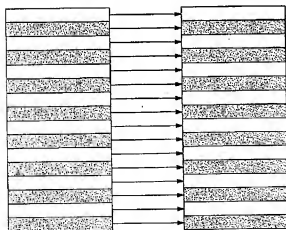
【図54b】



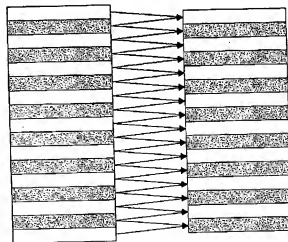
【図 63】



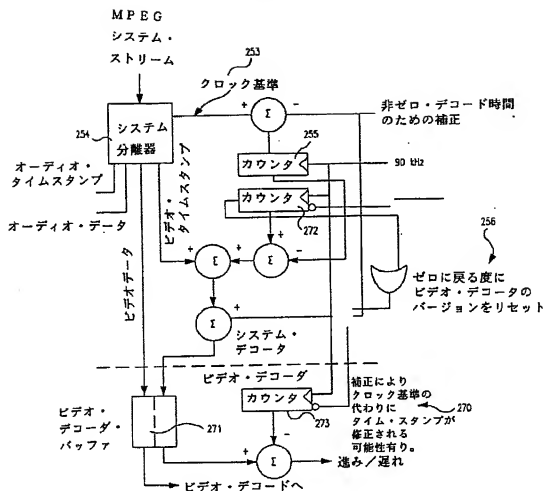
【図 71】



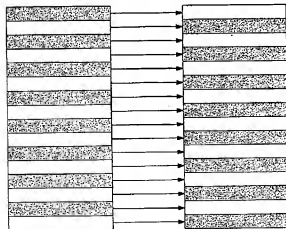
【図 72】



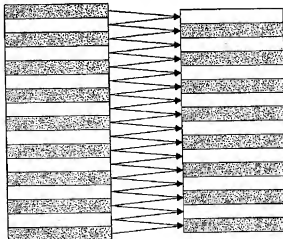
【図 6 4】



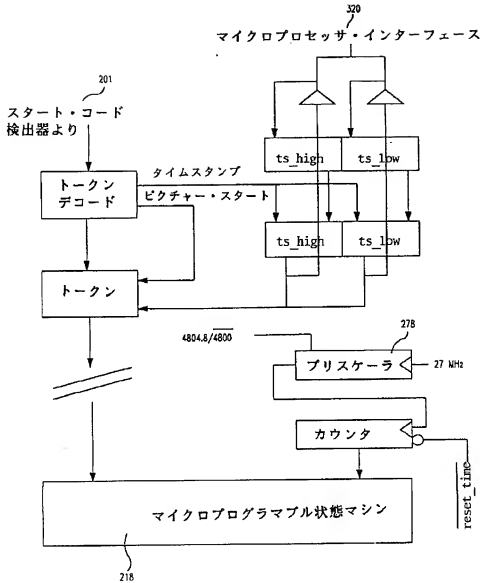
【図 7 3】



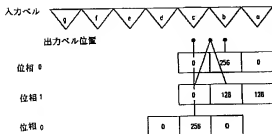
【図 7 4】



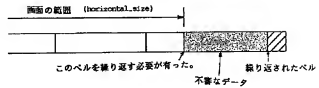
【図65】



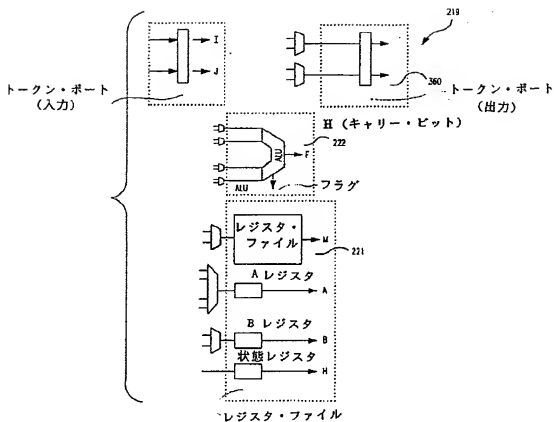
【図90】



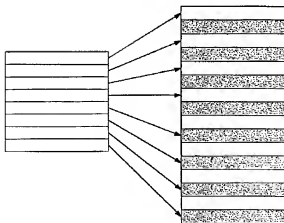
【図92】



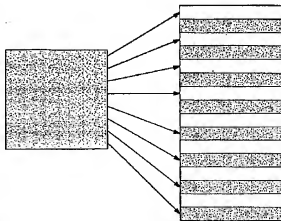
【図 67】



【図 75】



【図 77】





【図76】

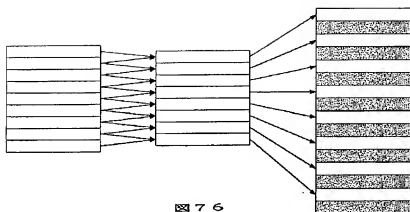


図 76

【図78】

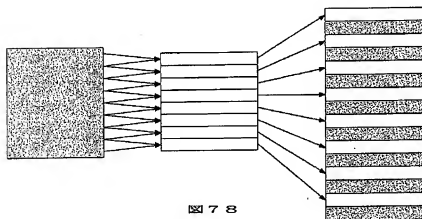
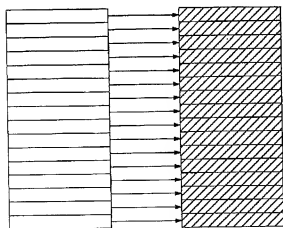
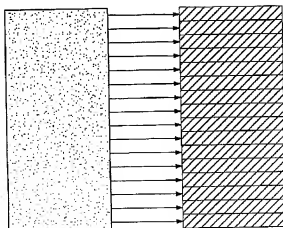


図 78

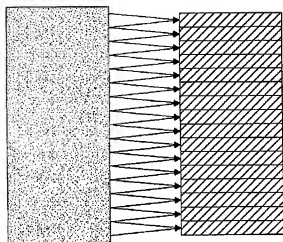
【図79】



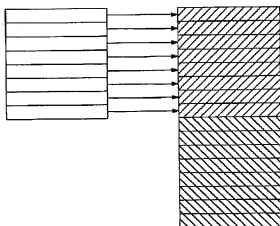
【図81】



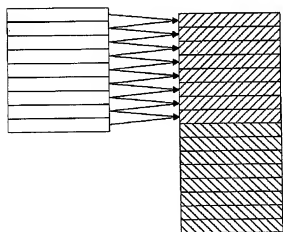
【図 82】



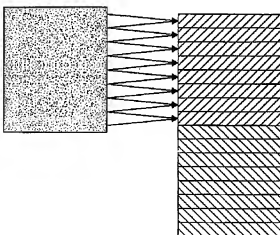
【図 83】



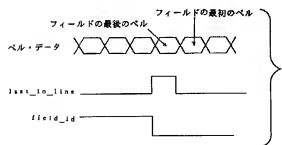
【図 84】



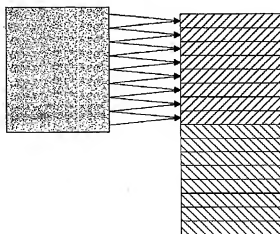
【図 85】



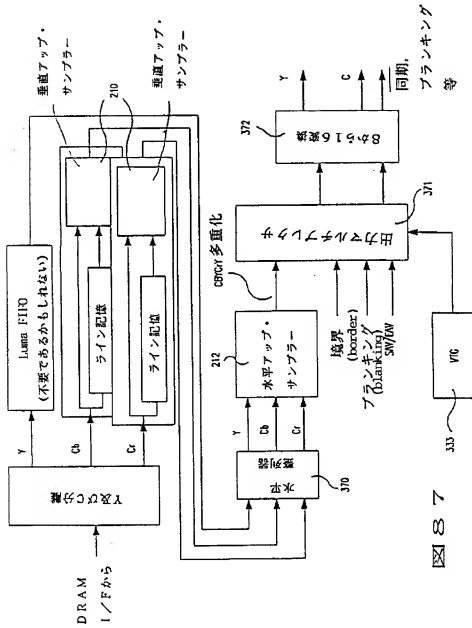
【図 93】



【図 86】



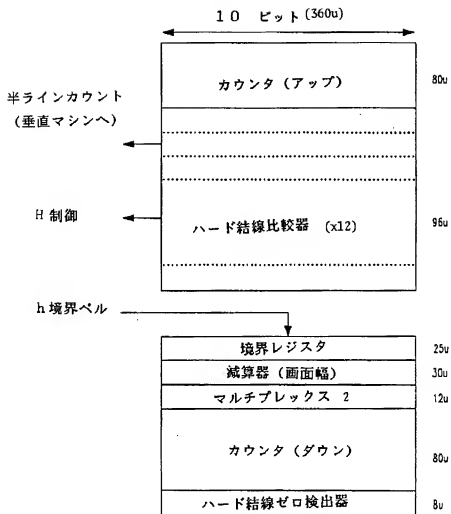
【図 87】



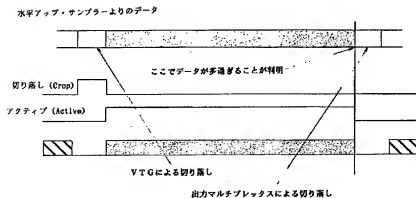




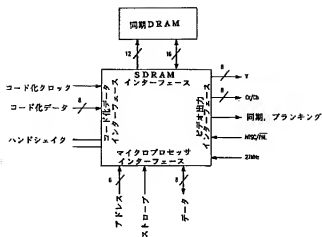
【図 97】



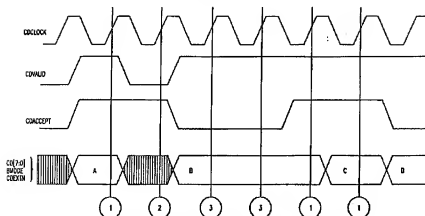
【図 99】



【図100】



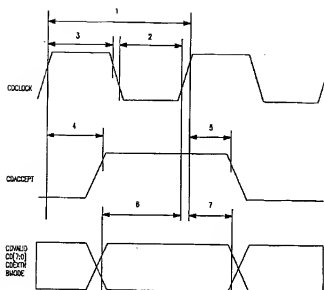
【図102】



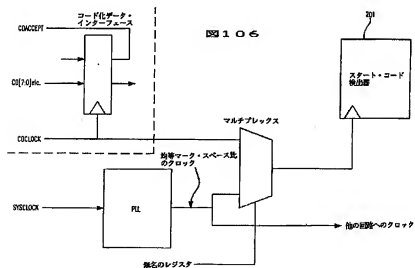
- 1) CDVALID及びCDACCEPTが共にハイであれば、データ転送が行われる。
- 2) CDVALIDがローであれば、システムはデータを削除する。
- 3) CDACCEPTがローであれば、システムはデータを受け入れることができません、データは受け入れられるまで待機される。

図 102

【図105】



【図106】





【図107】

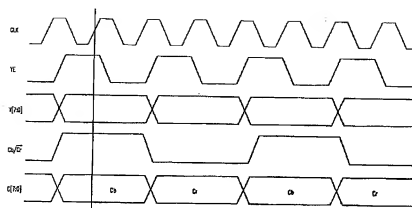


図107

【図108】

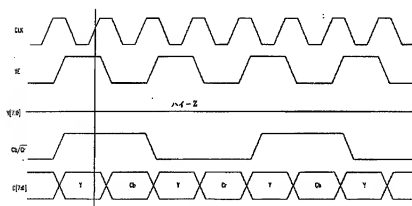


図108

【図109】

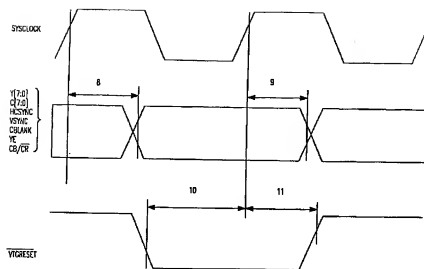


図 109

【図110】

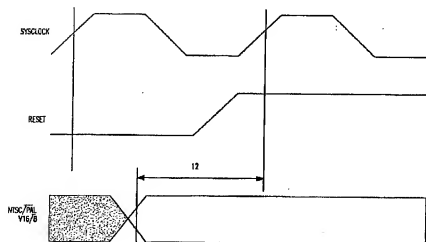


図 110



【☒ 1 1 2 a】

フィールド1

4		F V
5	フィールド同期	O I
6		O I
7		O I
8	イコライゼーション (等化)	O I
9		O I
10		F V
11		O I
12		O I
13		O I
14	プランキング	O I
15		O I
16		O I
17		O I
18		O I
19		O I
20		O O
21	アクティブ・ラインを 空にする。	F V
22		O O
23	1	O O
24	3	O O













イコライゼーション  
(等化)

## ブランキング

アクティブ・ラインを  
空にする。

1

3

261		477		0 0
262		479		0 0
263				0 1
264				0 1
265		イコライゼーション		1 1
266				0 0

c: CBLANKはアクティブであるけれどもY [7: 0]は  
尚ブランキングを出力する。

【図 1 1 2 b】

フィールド2

F V				F V
0 1	266			1 1
0 1	267	フィールド同期		1 1
0 1	268			1 1
0 1	269			1 1
0 1	270	イコライゼーション (等化)		1 1
0 1	271			1 1
0 1	272			1 1
0 1	273			1 1
0 1	274			1 1
0 1	275			1 1
0 1	276	ブランキング		1 1
0 1	277			1 1
0 1	278			1 1
0 1	279			1 1
0 1	280			1 1
0 1	281			1 1
0 0	282		0	1 0
0 0	283	アクティブ・ラインを 空にする。		1 0
0 0	284			1 0
0 0	285	0		1 0
0 0	286	2		1 0

イコライゼーション  
(等化)

## ブランキング

アクティブ・ラインを  
空にする。

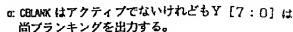
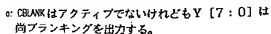
Q

2

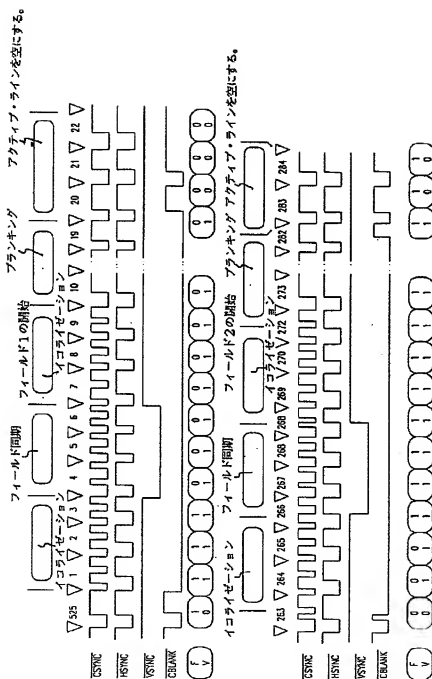
Figure 1 is a schematic diagram illustrating the experimental setup for studying the effect of the number of layers on the properties of the polymer film. The diagram shows a cross-section of a polymer film with a central layer labeled "イコライゼーション" (Isolation) and two outer layers labeled "478". The film is shown in three states: 1. Initial state (top), 2. Intermediate state (middle), and 3. Final state (bottom). The film is labeled with "523" and "524" on the left and "10" on the right.

α.CBLANKはアクティブであるけれどもY [7:0]は  
尚ブランキングを出力する。

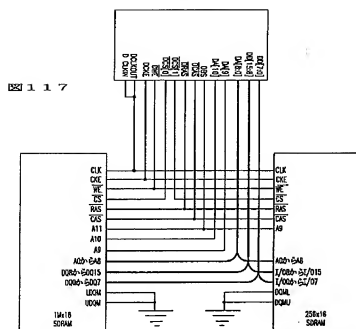
【図 1 1 3 b】



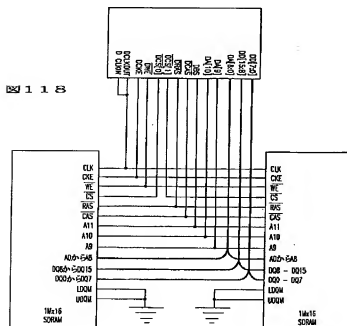
【図 114】



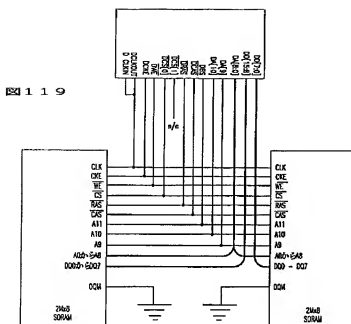
【図117】



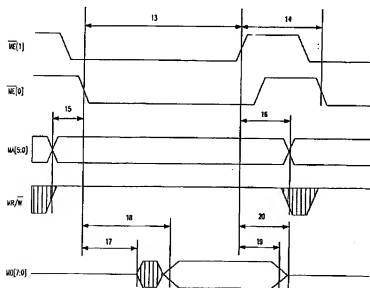
【図118】



【图 1 1 9】

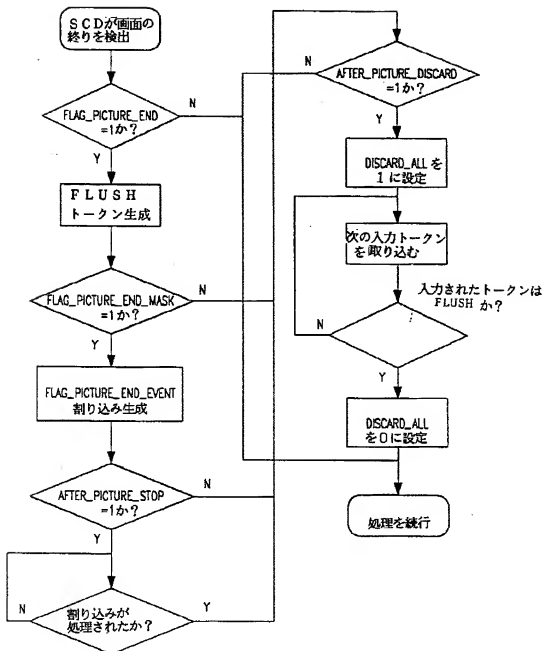


【图 1 2 3】





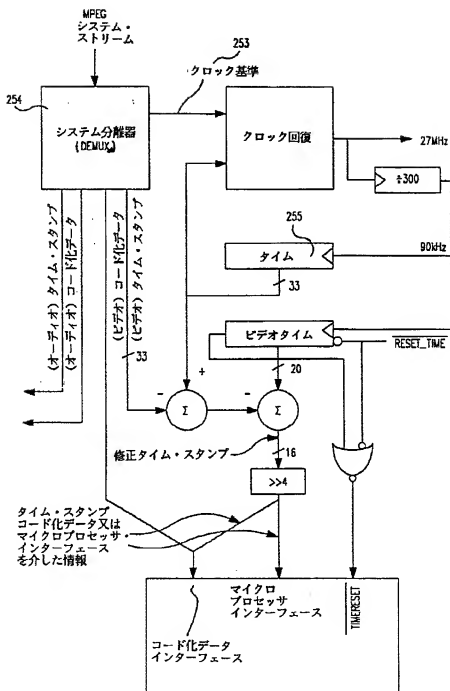
【図120】



```

graph TD
    Start([ゼロでない  
START_CODE_SEARCH  
を設定]) --> DiscardAll{DISCARD_ALL  
=1か?}
    DiscardAll -- Y --> FlushNext[次の入力トークン  
取り込み]
    DiscardAll -- N --> StartCodeSearch{START_CODE_SEARCH  
=0?}
    StartCodeSearch -- Y --> SetStartCode[START_CODE_SEARCH  
を0に設定]
    StartCodeSearch -- N --> EndSearchMask{END_SEARCH_MASK  
=1か?}
    EndSearchMask -- Y --> EndSearchEvent[END_SEARCH_EVENT  
割り込み生成]
    EndSearchMask -- N --> AfterSearchStop{AFTER_SEARCH_STOP  
=1か?}
    AfterSearchStop -- Y --> FlushProcessed{割り込みは  
処理されたか?}
    FlushProcessed -- Y --> End([処理を続行])
    FlushProcessed -- N --> StartCodeSearch
    AfterSearchStop -- N --> StartCodeSearch
    EndSearchEvent --> StartCodeSearch
    SetStartCode --> StartCodeSearch
    FlushNext --> DiscardAll
  
```

【図122】



【図 124】

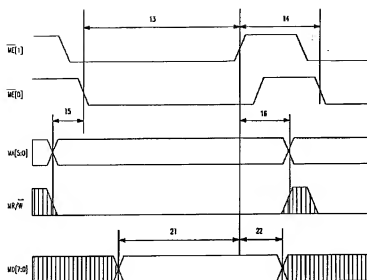


図 124

フロントページの続き

- (72)発明者 アンソニー マーク ジョーンズ  
イギリス国 ブリストル ビーエス17 5  
ティビーエイト テンブラーロード 31
- (72)発明者 マーティン ウィリアム ソザラン  
イギリス国 グロスターシャー ジーエル  
11 ジービーディ スティンチコン ウ  
ィックレイン ザライディングズ (番地  
なし)
- (72)発明者 コリン スミス  
イギリス国 ブリストル ビーエス18 4  
エックスエフ ビショップサトン パーク  
スフィールドガーデンス 34
- (72)発明者 ヘレン ローズマリー フィンチ  
イギリス国 グロスターシャー ジーエル  
12 7エヌディ ウットンアンダーエ  
ッジ クーン タイレイ (番地なし)
- (72)発明者 アンソニー ビーター ジョン クレイド  
ン  
イギリス国 エイボン ビーエー2 6 ビ  
ーゼット パース シドニービルディング  
ズ 14

- (72)発明者 ドナルド ウィリアム ダブリュ. パタ  
ーソン  
イギリス国 ブリストル ビーエス6 7  
ジェイダブリュ レッドランド プレニム  
ロード 12
- (72)発明者 マーク バーンズ  
イギリス国 ウィルトシャー エスエヌ14  
6エルビー チェンハム マイケル キ  
ングストンストリート ザクロズ 16
- (72)発明者 アンドリュ ビーター カリゴウスキ  
イギリス国 ブリストル ビーエス8 2  
ディビー クリフトン アルマロード 87  
ファーストフロアフラット
- (72)発明者 ウィリアム ビー. ロビンズ  
イギリス国 グロスターシャー ジーエル  
11 5ビーイー カム スプリングヒル  
19
- (72)発明者 ニコラス パーチ  
イギリス国 ブリストル ビーエス9 4  
ジェイビーヘンリーズ ブローンダウンア  
ベニュー 71

【外国語明細書】

## VIDEO DECOMPRESSION

This application claims priority under British Application Serial No. 9415413.5 filed July 29, 1994.

5       The present invention relates generally to a new and improved system for decoding a plurality of audio and video signals and, more particularly, to a new and improved system for decoding a plurality of MPEG audio and video signals.

10       A serial pipeline processing system of the present invention comprises a single two-wire bus used for carrying unique and specialized interactive interfacing tokens, in the form of control tokens and data tokens, to a plurality of adaptive decompression circuits and the like positioned as a reconfigurable pipeline processor.

15       United States Patent 5,111,292 discloses an apparatus for encoding/decoding a HDTV signal for e.g. terrestrial transmission includes a priority selection processor for parsing compressed video codewords between high and low priority channels for transmission. A compression circuit responsive to high definition video source signals provides hierarchically layered codewords CW representing compressed video data and associated  
20       codewords T, defining the types of data represented by codewords CW. The priority selection processor, responsive to the codewords CW and T, counts the number of bits in predetermined blocks of data and determines the number of bits in each block to be allocated to the respective channels. Thereafter the processor parses the codewords CW into high and low priority  
25       codeword sequences wherein the high and low priority codeword sequences correspond to compressed video data of relatively greater and lesser importance to image reproduction respectively.

      United States Patent No. 5,122,875 discloses an apparatus for encoding/decoding an HDTV signal. The apparatus includes a compression  
30       circuit responsive to high definition video source signals for providing hierarchically layered codewords CW representing compressed video data and associated codewords T, defining the types of data represented by the codewords CW. A priority selection circuit, responsive to the codewords CW and T, parses the codewords CW into high and low priority codeword sequences wherein the high and low priority codeword sequences correspond to  
35       compressed video data of relatively greater and lesser importance to image reproduction respectively. A transport processor, responsive to the high and

low priority codeword sequences, forms high and low priority transport blocks of high and low priority codewords, respectively. Each transport block includes a header, codewords CW and error detection check bits. The respective transport blocks are applied to a forward error check circuit for applying additional error check data. Thereafter, the high and low priority data are applied to a modem wherein quadrature amplitude modulates respective carriers for transmission.

Accordingly, those skilled in the art have recognized a long felt need for a new and improved video decompression system obviating the deficiencies of the prior art systems. The present invention clearly fulfills this need.

Figure 1 illustrates data flow through a preferred embodiment in the present invention;

Figure 2 shows an example of a 13 bit word used to address 8 bit data in a 64 X 32 RAM;

Figure 3 is a functional block diagram of a Register file in the present invention;

Figure 4 illustrates data flow in a register file as shown in Figure 3;

Figure 5 is a block diagram illustrating register file address decoding. In accordance with the present invention;

Figure 6 is a block diagram of a Microcodable State Machine, in accordance with the present invention;

Figure 7 shows a fixed width word, in accordance with the present invention, used for addressing and having an address field, a substitution field and a substitution header;

Figure 8 is a block diagram of one example of an Arithmetic Core in accordance with the present invention;

Figure 9 illustrates the basis steps in a method, in accordance with the present invention, for performing an IDCT on input data;

Figure 10 is a block diagram illustrating the combined, simplified, two-stage architecture of an IDCT system, in accordance with the present invention;

Figure 11 is a simplified block diagram of an integrated circuit that comprises the main system components of the IDCT shown in Figure 10;

Figure 12a and Figure 12b taken together are a block diagram of a pre-processing circuit corresponding to one of the main system component; for ease of explanation, these figures are referred collectively as Figure 12;

## 3

Figure 13a, Figure 13b and Figure 13c depict timing diagrams which illustrate the relationships between timing and control signals in the IDCT system of a preferred embodiment;

5 Figure 14a and Figure 14b taken together are a block diagram of a common processing circuit in the IDCT system; for ease of explanation, these figures are referred to collectively as Figure 14;

Figure 15a, Figure 15b, Figure 15c and Figure 15d taken together are a block diagram of a post-processing circuit which corresponds to another main component of the system and are referred collectively as Figure 15;

10 Figure 16 is a block diagram, in accordance with the present invention illustrating an IDCT having a twin data stream, a transpose RAM and an improved buffer;

Figure 17 is a block diagram showing in further detail the 1-D IDCT system shown in Figure 16;

15 Figure 18 is a block diagram showing greater detail of the transform system as shown in Figure 17;

Figure 19 is a block diagram showing in greater detail the input buffer shown in Figure 18;

20 Figure 20 is a simplified block diagram of a pre-processing circuit "PREC", in accordance with the present invention;

Figure 21 is a block diagram illustrating a common processing circuit "CBLK" found in the IDCT;

Figure 22 is a block diagram of a post-processing circuit "POSTC";

25 Figure 23 is another illustration of the post-processing circuit shown in Figure 22;

Figure 24 is a block diagram depicting a round and saturate block, in accordance with the present invention;

Figure 25 is a block diagram of an output buffer in the present invention;

30 Figure 26 is a block diagram of a control shift register, in accordance with the present invention;

Figure 27 is a block diagram of a control shift register decode in the present invention;

Figure 28 depicts a control shift register and an input control buffer;

35 Figure 29 illustrates a control circuit for a T2 data stream;

Figure 30 shows data in a counter for a T1 data stream;

Figure 31 depicts data in a counter for a T2 data stream in the present invention;

Figure 32 is a timing diagram showing the initialization of the IDCT and associated circuitry

5      Figure 33 is a timing diagram showing the interleaving of T1 and T2 data;

Figure 34 is a timing diagram illustrating slippage and recovery of T2 data;

10      Figure 35 is a timing diagram depicting a flushing operation of the IDCT and associated circuitry in the present invention;

Figure 36 illustrates start-up of the system, in accordance with the present invention;

Figure 37 depicts slippage and recovery in the early stages of interleaving T1 and T2 data;

15      Figure 38 illustrates another preferred embodiment of the IDCT system shown in Figures 16 through 37;

Figure 39 shows MPEG information streams being demultiplexed, in accordance with the present invention, into elementary streams containing date and timestamp information;

20      Figure 40 depicts a first embodiment of an elementary stream timestamp error determination and time synchronization system, in accordance with the present invention;

Figure 41 illustrates a second embodiment of an elementary stream timestamp error determination and time synchronization system, in accordance with the present invention;

25      Figure 42 shows a third embodiment of an elementary stream timestamp error determination and time synchronization system, in accordance with the present invention;

Figure 43 depicts a first embodiment of a video timestamp error determination and time synchronization system, in accordance with the present invention;

30      Figure 44 illustrates a second embodiment of a video timestamp error determination and time synchronization system, in accordance with the present invention;

35      Figure 45 shows the second embodiment of a video timestamp error determination and time synchronization system as shown in Figure 44 and operating at 30Hz;



## 5

Figure 46 shows timestamp information flow through the system of the present invention;

Figure 47 is a block diagram illustrating synchronization time information being processed by a microprogrammable state machine;

5 Figure 48 is a block diagram illustrating a first preferred embodiment of the present invention;

Figure 49 is another block diagram illustrating the first preferred embodiment of the present invention;

10 Figure 50 depicts a second preferred embodiment of the present invention;

Figure 51 illustrates a detailed method of addressing used by the second preferred embodiment, in accordance with the present invention;

Figure 52 is a block diagram showing an apparatus for decoding Huffman VLCs, in accordance with the present invention;

15 Figure 53 is a schematic diagram showing the overall structure of the parallel huffman decoder of the present invention;

Figure 54 is a schematic diagram illustrating a ROM adapted for decoding parallel huffman codes;

20 Figure 55 illustrates a first embodiment of a ROM adapted for decoding parallel huffman codes;

Figure 56 illustrates a second embodiment of a ROM adapted for decoding parallel huffman codes;

Figure 57 depicts a third embodiment of a ROM adapted for decoding parallel huffman codes;

25 Figure 58 is a block diagram illustrating the primary system component of one embodiment of the present invention;

Figure 59 is a block diagram depicting the start code detector of the present invention;

30 Figure 60 is a block diagram showing the parser of the present invention;

Figure 61 is a block diagram depicting the primary components of the spatial processing circuitry of the present invention;

Figure 62 is a block diagram illustrating the display circuitry, in accordance with the present invention;

35 Figure 63 illustrates one embodiment of timestamp management, in accordance with the present invention;

Figure 64 shows another embodiment of timestamp management in the present invention;

Figure 65 is a block diagram depicting the hardware components of the system of the present invention;

5 Figure 66 is a block diagram providing an overview of the system components of the microcontroller of the present invention;

Figure 67 is a simplified diagram illustrating the Arithmetic core of the present invention;

Figure 68 illustrates the ALU of the present invention;

10 Figure 69 depicts a register file, in accordance with the present invention;

Figure 70 illustrates the writing to independent bus registers in the present invention;

Figure 71 illustrates frame-based prediction wherein  $\text{vector}[1] = 0$  and  $\text{vector}[0] = 0$ ;

15 Figure 72 depicts frame-based prediction wherein  $\text{vector}[1] = 0$  and  $\text{vector}[0] = 1$ ;

Figure 73 shows frame-based prediction wherein  $\text{vector}[1] = 1$  and  $\text{vector}[0] = 0$ ;

20 Figure 74 illustrates frame-based prediction wherein  $\text{vector}[1] = 1$  and  $\text{vector}[0] = 1$ ;

Figure 75 depicts field-based prediction wherein  $\text{motion\_vertical\_field\_select} = 0$  and  $\text{vector}[0] = 0$ ;

25 Figure 76 illustrates field-based prediction wherein  $\text{motion\_vertical\_field\_select} = 0$  and  $\text{vector}[0] = 1$ ;

Figure 77 similarly illustrates field-based prediction wherein  $\text{motion\_vertical\_field\_select} = 1$  and  $\text{vector}[0] = 0$ ;

Figure 78 shows field-based prediction wherein  $\text{motion\_vertical\_field\_select} = 1$  and  $\text{vector}[0] = 1$ ;

30 Figure 79 shows field-based prediction in frame pictures wherein  $\text{motion\_vertical\_field\_select} = 0$  and  $\text{vector}[0] = 0$ ;

Figure 80 illustrates the prediction of Figure 79 wherein  $\text{motion\_vertical\_field\_select} = 0$  and  $\text{vector}[0] = 1$ ;

35 Figure 81 shows the prediction mode of Figure 79 wherein  $\text{motion\_vertical\_field\_select} = 1$  and  $\text{vector}[0] = 0$ ;

Figure 82 shows the prediction mode of Figure 79 wherein both  $\text{motion\_vertical\_field\_select}$  and  $\text{vector}[0] = 1$ ;

Figure 83 illustrates an additional mode of prediction filtering;

Figure 84 shows still another prediction mode;

Figure 85 illustrates yet another prediction mode, in accordance with the present invention;

5 Figure 86 shows another prediction mode of the present invention;

Figure 87 is a block diagram illustrating the organization of the various system components of the display system of the present invention;

Figure 88 depicts a 4:3 filtering operation;

Figure 89 depicts a 3:2 filtering operation;

10 Figure 90 illustrates a 2:1 filtering operation of the present invention;

Figure 91 shows a three tap filter used in the present invention;

Figure 92 illustrates the repetition of erroneous pixels;

Figure 93 depicts the field\_id signal of the present invention;

15 Figure 94 shows the horizontal timing points (cycles), in accordance with the present invention;

Figure 95 illustrates the PAL vertical timing at 625 lines per field, in accordance with the present invention;

Figure 96 illustrates the NTSCV vertical timing at 525 lines per field, in accordance with the present invention;

20 Figure 97 shows a horizontal counting machine, in accordance with the present invention;

Figure 98 illustrates border generation in the present invention;

Figure 99 depicts picture cropping, in accordance with the present invention;

25 Figure 100 is a block diagram illustrating the present invention as a chip;

Figure 101 illustrates the sysclock requirements of the present invention;

30 Figure 102 depicts the two-wire protocol on a coded data interface, in accordance with the present invention;

Figure 103 shows a DATA token of the present invention;

Figure 104 shows a FLUSH token of the present invention;

Figure 105 illustrates the timing of the coded data interface;

35 Figure 106 depicts using non-even mark-space ratio CDCLOCK, in accordance with the present invention; -

Figure 107 shows output timing in 16 bit mode in the present invention;

Figure 108 illustrates output timing in 8 bit mode in the present invention;

Figure 109 shows the timing of the video output interface in the present invention;

5 Figure 110 depicts video output mode signals, in accordance with the present invention;

Figure 111 shows horizontal timing in the present invention;

Figure 112 shows the vertical timing for a 525 line system;

Figure 113 depicts the vertical timing for a 625 line system;

10 Figure 114 illustrates the sync and blanking signals for a 525 line system, in accordance with the present invention;

Figure 115 shows the sync and blanking signals for a 625 line system, in accordance with the present invention;

15 Figure 116 illustrates a zero SDRAM connection configuration in the present invention;

Figure 117 shows one SDRAM connection configuration in the present invention;

Figure 118 depicts a two SDRAM connection configuration, in accordance with the present invention;

20 Figure 119 illustrates a three SDRAM connection configuration

Figure 120 is a flow chart depicting the flag\_picture\_end operation, in accordance with the present invention;

Figure 121 is a flow chart showing the start\_code\_search operation, in accordance with the present invention;

25 Figure 122 shows timestamp modification, in accordance with the present invention

Figure 123 illustrates the read timing for the microprocessor interface; and

Figure 124 shows the write timing for the microprocessor interface.

30 In the ensuing description of the practice of the invention, the following terms are frequently used and are generally defined by the following glossary:

## GLOSSARY

- BLOCK:** An 8-row by 8-column matrix of pels, or 64 DCT coefficients (source, quantized or dequantized).
- CHROMINANCE (COMPONENT):** A matrix, block or single pel representing one of the two color difference signals related to the primary colors in the manner defined in the bit stream. The symbols used for the color difference signals are Cr and Cb.
- CODED REPRESENTATION:** A data element as represented in its encoded form.
- CODED VIDEO BIT STREAM:** A coded representation of a series of one or more pictures as defined in this specification.
- CODED ORDER:** The order in which the pictures are transmitted and decoded. This order is not necessarily the same as the display order.
- COMPONENT:** A matrix, block or single pel from one of the three matrices (luminance and two chrominance) that make up a picture.
- COMPRESSION:** Reduction in the number of bits used to represent an item of data.
- DECODER:** An embodiment of a decoding process.
- DECODING (PROCESS):** The process defined in this specification that reads an input coded bitstream and produces decoded pictures or audio samples.
- DISPLAY ORDER:** The order in which the decoded pictures are displayed. Typically, this is the same order in which they were presented at the input of the encoder.
- ENCODING (PROCESS):** A process, not specified in this specification, that reads a stream of input pictures or audio samples and produces a valid coded bitstream as defined in this specification.
- INTRA CODING:** Coding of a macroblock or picture that uses information only from that macroblock or picture.
- LUMINANCE (COMPONENT):** A matrix, block or single pel representing a monochrome representation of the signal and related to the primary colors in the manner defined in the bit stream. The symbol used for luminance is Y.
- MACROBLOCK:** The four 8 by 8 blocks of luminance data and the two (for 4:2:0 chroma format) four (for 4:2:2 chroma format) or eight (for 4:4:4 chroma format) corresponding 8 by 8 blocks of chrominance data coming from a 16 by 16 section of the luminance component of the picture. Macroblock is sometimes used to refer to the pel data and sometimes to the coded representation of the pel values and other data elements defined in the macroblock header of

the syntax defined in this part of this specification. To one of ordinary skill in the art, the usage is clear from the context.

**MOTION COMPENSATION:** The use of motion vectors to improve the efficiency of the prediction of pel values. The prediction uses motion vectors to provide offsets into the past and/or future reference pictures containing previously decoded pel values that are used to form the prediction error signal.

**MOTION VECTOR:** A two-dimensional vector used for motion compensation that provides an offset from the coordinate position in the current picture to the coordinates in a reference picture.

**NON-INTRA CODING:** Coding of a macroblock or picture that uses information both from itself and from macroblocks and pictures occurring at other times.

**PEL:** Picture element.

**PICTURE:** Source, coded or reconstructed image data. A source or reconstructed picture consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. For progressive video, a picture is identical to a frame, while for interlaced video, a picture can refer to a frame, or the top field or the bottom field of the frame depending on the context.

**PREDICTION:** The use of a predictor to provide an estimate of the pel value or data element currently being decoded.

**RECONFIGURABLE PROCESS STAGE (RPS):** A stage, which in response to a recognized token, reconfigures itself to perform various operations.

**SLICE:** A series of macroblocks.

**TOKEN:** A universal adaptation unit in the form of an interactive interfacing messenger package for control and/or data functions.

**START CODES [SYSTEM AND VIDEO]:** 32-bit codes embedded in a coded bitstream that are unique. They are used for several purposes including identifying some of the structures in the coding syntax.

**VARIABLE LENGTH CODING; VLC:** A reversible procedure for coding that assigns shorter code-words to frequent events and longer code-words to less frequent events.

**VIDEO SEQUENCE:** A series of one or more pictures.

Briefly, and in general terms, the present invention provides a new and improved method and apparatus particularly adapted for use in a two-wire pipeline system having various control and DATA tokens. The major ele-

11

ments of the system may include a Start Code Detector, a Video Parser incorporating a Huffman Decoder and a Microprogrammable State Machine (MSM), an Inverse Discrete Cosine Transform (IDCT), a synchronous DRAM controller with an associated address generation unit, appropriate prediction circuitry and display circuitry which includes upsampling and video timing generation.

More importantly, various embodiments of the invention may include an MPEG video decompression method and apparatus utilizing a plurality of stages interconnected by a two-wire interface arranged as a pipeline processing machine. Control tokens and DATA Tokens pass over the single two-wire interface for carrying both control and data in token format. A token decoder circuit is positioned in certain of the stages for recognizing certain of the tokens as control tokens pertinent to that stage and for passing unrecognized control tokens along the pipeline. Reconfiguration processing circuits are positioned in selected stages and are responsive to a recognized control token for reconfiguring such stage to handle an identified DATA Token. A wide variety of unique supporting subsystem circuitry and processing techniques are disclosed for implementing the system, including memory addressing, transforming data using a common processing block, time synchronization, asynchronous swing buffering, storing of video information, a parallel Huffman decoder, and the like.

By way of example, and not necessarily by way of limitation, the present invention may include among its various features an apparatus for synchronizing time having, a time stamp for determining presentation time, a clock reference for initializing system time in a first circuit, a first time counter in communication with the clock reference for keeping system time in a first circuit and a second time counter initialized by the clock reference in a second circuit synchronized with the first time counter, for keeping a local copy of the system time and for determining the presentation timing error between the local copy of system time and system time by comparing the time stamp to the second time counter. It further includes an apparatus for synchronizing a system decoder and a video decoder using a time stamp for determining display time, a clock reference for initializing system time in the system decoder, a first time counter in communication with the clock reference for keeping system time in the system decoder and a second time counter initialized by the clock reference in the video decoder synchronized with the first time counter, for keeping a local copy of system time and for determining

the display timing error between the local copy of system time and system time by comparing the time stamp to the second time counter.

Still another embodiment of the invention includes an apparatus for synchronizing a first circuit and a second circuit using a clock reference for initializing system time in the first circuit, a first circuit having a time counter in  
5 communication with the clock reference for keeping system time, a first elementary stream time counter in the first circuit for providing elementary stream time. The first circuit is adapted to receive a time stamp, and the first circuit generates synchronization time by adding elementary stream time to  
10 the time stamp and subtracting system time. The second circuit is adapted to receive synchronization time from the first circuit and has a second elementary stream time counter in synchronization with the first elementary stream time counter for providing a local copy of the elementary stream time and for  
15 determining a timing error between the system time and the time stamp by comparing synchronization time to the local copy of elementary stream time. In this way, the clock reference signal does not have to be passed directly to the second circuit in order to determine the timing error.

In another embodiment of the invention, an apparatus for synchronizing a first circuit and a second circuit has a clock reference for initializing  
20 system time in the first circuit. The first circuit has a time counter in communication with the clock reference for keeping system time, and a first video time counter for providing video decoding time. The first circuit is adapted to receive a video time stamp and subtracting system time. The second circuit is  
25 adapted to receive synchronization time from the first circuit and has a second video time counter in synchronization with the first video time counter for providing a local copy of video decoding time and for determining a timing error between system time and the video time stamp by comparing synchronization time to the local copy of video decoding time. Accordingly, the clock  
30 reference signal does not have to be passed directly to the second circuit in order to determine the timing error.

The present invention also includes a method for providing timing information by providing a video data stream having a time stamp carried in packet header wherein the time stamp refers to the first picture in the packet  
35 of data. In the next step a register is provided having a flag used to indicate valid time stamp information which is taken from the packet header and placed into the register. Next, the time stamp is removed from the video data stream and placed in the register. Next, the method encounters a picture



start and subsequently examines the status of the register to determine if valid time stamp information is contained in the register by checking the flag status. A time stamp is generated in response to the picture start if the flag indicates valid time stamp information is contained in the register and then the time stamp is inserted back into the data stream.

Another embodiment of the invention includes an apparatus described above wherein the elementary stream time counters are restricted to 16 bits.

Likewise, there is an apparatus as described above, wherein the second elementary stream time counter located in the elementary stream decoder is restricted to 16 bits. Furthermore, there is an apparatus as described above wherein the synchronization time is restricted to 16 bits for controlling the elementary stream decoder.

The present invention also has a process for decoding video and for determining display time errors against a threshold value. It then parses video data into tokens for further processing, determining if a time stamp token is indicated, comparing the time stamp token to a video time, and generates a compared value to determine an indicative of timing error. Next, it determines whether the compared value, when compared against a threshold value, is within acceptable parameters when a timing error is indicated and indicates when the compared value is outside acceptable parameters.

An alternative embodiment of the invention includes an apparatus for using a system decoder and a video decoder. The system decoder is adapted to accept MPEG system streams and demultiplexing video data and the video time stamp from the stream. The system decoder has a first time counter representative of system time. The video decoder accepts the video data and the video time stamp, and has a second time counter in synchronization with the first time counter. The video decoder also has a decoder buffer for accepting the video data at a substantially constant rate and outputting the video data at a varying rate and for passing a video time stamp. The video decoder while decoding a picture from the video data also compares the video time stamp for the decoded picture with the second time counter to determine the appropriate display time. There is also a method for determining a timing error between a first circuit and a second circuit by providing the first circuit with a system time (SY), a time stamp (TS), and an elementary stream time (ET), obtaining synchronization time (X) by using the elementary stream time (ET), the time stamp (TS), and the system time (SY), in accordance with the equation  $X=ET + TS-SY$ , providing synchronization

time (X) to the second circuit and generating a synchronized elementary stream time (ET2) and obtaining a timing error by using synchronized time (X) and in accordance with the equation  $ET2-X$ . Hence, the first circuit can be time synchronized with the second circuit without passing system time to the second circuit.

Another method for determining a timing error between a first circuit and a second circuit has the following steps: providing the first circuit with a time stamp (TS), and an initial time (IT), obtaining a synchronization time (X) by using the time stamp (TS) and the initial time (IT), in accordance with the equation  $X=TS-1$ , providing synchronization time (X) to the second circuit and generating a synchronized elementary stream time (ET) and obtaining a timing error by using synchronized time (X) and in accordance with the equation  $ET-X$ . In this way, the first circuit can be time synchronized with the second circuit without passing system time to the second circuit.

Still another method for determining a timing error between a first circuit and a second circuit includes the following steps: providing the first circuit with a system time (SY), a video time stamp (VTS), and a video decoding time (VT), obtaining synchronization time (X) by using the video decoding time (VT), the video time stamp (VTS) and the system time (SY). In accordance with the equation  $X=VT+VTS-SY$ , providing synchronization time (X) to the second circuit and generating a video decoding time (VT2) in the second circuit which is synchronized to the video decoding time (VT) in the first circuit, and obtaining a time error by using synchronized time (X) and in accordance with the equation  $VT2-X$ . Accordingly, the first circuit can be time synchronized with the second circuit without passing system time to the second circuit.

In accordance with the present invention, the parallel Huffman decoder block will decode MPEG Huffman coded Variable Length Codes (VLCs) and Fixed Length Codes (FLCs), and pass through tokens under the control of the parser microprogrammable state Machine (MSM), and can sustain a high throughput.

In one embodiment of the invention a code lookup technique is employed to decode Huffman codes to achieve performance requirements and to handle the second MPEG-2 transform coefficient table which is irregular or non-canonical in nature. Practice of the invention also facilitates decoding certain more complex components from the stream in a single cycle without the assistance of an external controller. Examples of such complex

components are Escape-coded coefficients, Intra-DC values and Motion Vector deltas, all of which are present in the stream as combined VLC/FLC components.

To decode a VLC, input is first loaded into the two input data registers handling most significant and least significant data. A selector is used to align the beginning of the next VLC with the ROM input. Hence, for a very first VLC, the selector outputs the top 28 bits of its 59-bit input and the top 16 bits of these are passed to a Huffman Code ROM. For subsequent VLCS, the selector effectively shifts the input according to the total count of bits decoded thus far, the count is maintained by adding the size of each VLC, as it is decoded, to a running total. The various word widths are a result of the maximum coded size which can be decoded, which is the 28-bit MPEG-1 Escape Coded Coefficient, and the maximum VLC size which is 16 bits (DCT coefficient tables).

The "table select" input is used to select between the various different Huffman code tables required by MPEG.

The ROM has addresses which are controlled with a selector/shifter. The ROM performs a VLC table index calculation, followed by the index-to-data operation that yields decoded data.

The index calculation is a content addressable memory (CAM) operation with "don't care" matching implemented to handle the Huffman codes which form the presented data. Since the index generation is performed in a look-up manner (rather than algorithmically) there is no restriction to handling tables which are canonical.

The ROM address of the present invention is in two fields. The larger field is the bit-pattern to be decoded, and the smaller field selects which Huffman code table is to be examined. In addition to the complete MPEG code tables, the ROM also has entries to identify illegal VLC patterns, which exist for some code tables.

In another embodiment of the invention, a procedure is used for providing a word with fixed width, having a fixed number of bits to be used for addressing variable width data, and having a width defining field and address field. There is also a procedure for addressing memory with a fixed width word, having a fixed number of bits, to be used for addressing data and having a substitution field and an address field, and an apparatus for addressing memory, including a state machine and an arithmetic core.

The procedure for addressing memory is characterized by providing a fixed width word having a predetermined fixed number of bits to be used for addressing variable width data, defining the fixed width word with a width defining field and an address field, providing the width defining field with at least one bit to serve as the termination marker, defining the address field with a plurality of bits defining the address of data, varying the size of bits in the address field in inverse relation to the size of the variable width data, varying the number of bits in the width defining field in direct relation to the size of the variable width data, and maintaining a fixed width word for addressing variable width data while varying the width of the width defining field and the address field.

The procedure for addressing memory may also include defining the address field with a plurality of bits defining the address of the data, defining a variable width substitution field with a least one substitution bit, the substitution field having at least one bit to serve as a termination marker between the address field and the substitution field, using the substitution field to indicate substituted bits from a separate addressing source, and maintaining a fixed width word for addressing variable width data while inversely varying the width of the address field and the width of the substitution field.

In accordance with the invention, a process for addressing variable width data in a memory may be characterized by providing a memory having words of predetermined width and composed of partial words, rotating the partial word to be accessed to a least significant bit justification, extending the remaining part of the word so that the accessed word will be recognized as a partial word, restoring the remaining part of the word, and rotating the word until the partial word is restored to its original position.

The invention may also include a method and apparatus for addressing memory wherein a word is provided with fixed width, having a fixed number of bits to be used for addressing variable width data, and having a width defining field and address field. In addition, a procedure for addressing memory with a fixed width word, having a fixed number of bits, to be used for addressing data and having a substitution field and an address field, may be used.

The invention may also include a method of accessing from RAM a number M of words that is less than the predetermined fixed burst length N of the RAM, the RAM including an enable line that selectably enables and

17

disables reading from and writing to the RAM, the method comprising the steps of:

ordering N words to be read from or written to the RAM;  
determining when M words have been read from or written to the  
RAM, M being less than N; and

disabling the RAM upon determining M words had been read from or  
written to the RAM.

The invention may also include a method of accessing Dynamic  
Random Access Memory (DRAM) to store and retrieve data words associated  
with a two dimensional image, the DRAM including two separate banks, each  
bank being capable of operating a page mode to read and write the data  
words, the two dimensional image being organized in a two dimensional grid  
pattern of cells, each cell containing an M by N matrix of pixels, and the words  
associated with each cell occupying one page or less of a bank, the method  
comprising the steps of:

- (a) assigning each cell a particular one of the two banks so that  
all data words associated with that particular cell are read  
from and written to one particular page of that particular bank,  
the assignment of banks to cells being done such that each  
cell is associated with a different bank than any bordering cell  
which is also either in the same row or in the same column;
- (b) reading the data words associated with a cell that is com-  
posed of a matrix of pixels, and that is not aligned with the  
two dimensional grid pattern, but that is aligned with pixels in  
cells in the two dimensional grid pattern.
- (c) identifying which cells in the two dimensional grid pattern  
contain data words associated with the unaligned cell;
- (d) reading, from the first bank of DRAM, the data words associ-  
ated with one of the cells in the grid pattern identified as  
containing data words associated with the unaligned cell;
- (e) reading, from the second bank of DRAM, the data words  
associated with another of the cells in the grid pattern identi-  
fied as containing data words associated with the unaligned  
cell;
- (f) repeating steps (d) and (e) until all the data words associated  
with the unaligned cell have been read.

18

The invention may also provide a RAM interface for connecting a bus to RAM wherein a separate address generator generates the addresses the RAM interface needs to address the RAM. The address generator communicates with the RAM interface via a two-wire interface.

5 The invention may also include a method to control the buffering of encoded video data organized as frames or fields. This method involves determining the picture number of each incoming decoded frame, determining the expected presentation number at any time and marking any buffer as ready when its picture number is on or after the presentation number.

10 Accordingly, those concerned with the design, development, and utilization of systems for decoding video data have long recognized the need for enhanced performance as accomplished by the various features of the present invention. Other objects and advantages of the present invention will become apparent from the following more detailed description taken in  
15 conjunction with the accompanying drawings.

## DETAILED DESCRIPTIONS

The forthcoming "Detailed Description of the Invention" contains the following Sections:

- 1) Detailed Description of the Invention for Memory Address-  
5 ing  
Variable Length Fields Within a Fixed Width Word  
Using Fixed Width Word with Variable Length Fields to Perform  
Address Substitution  
Addressing Variable Width Data with a Fixed Width Word
- 10 2) Detailed Description of the Invention for Transforming Data  
using a Common Processing Block  
Theoretical Background of the Invention
- 15 3) Detailed Description of Invention for Time  
Synchronization
- 4) Detailed Description of the Invention for Asynchronous Swing  
Buffering
- 20 5) Detailed Description of the Invention for Storing Video Informa-  
tion
- 6) Detailed Description of the Invention for a Parallel Huffman  
Decoder
- 25 The Huffman Code ROM  
Maximizing Throughput  
FLCs and Tokens  
Implementation
- 7) MORE DETAILED DESCRIPTION

## DETAILED DESCRIPTION OF THE INVENTION

As an introduction to the illustrative embodiment(s) of the most general features of the invention, and referring more particularly to Figure 1 of the drawings, the data flow through the preferred embodiment 200 of the invention is shown. The embodiment of the present invention is preferably implemented using a two-wire pipeline system having various control and DATA tokens. The major elements of the system are a Start Code Detector 201, a Video Parser 202 incorporating a Huffman Decoder 203 and a Microprogrammable State Machine (MSM) 204, an Inverse Discrete Cosine Transform (IDCT) 205, a synchronous DRAM controller 206 with an associated address generation unit 207, appropriate prediction circuitry 208 and display circuitry 209 which includes upsampling 210 and 211 and video timing generation 212.

This application relates to similar subject matter disclosed in British Patent Application number 9405914.4 entitled "Video Decompression" filed on March 24, 1994, by Discovision Associates, and the latter application is specifically incorporated by reference in this application.

In accordance with the above, specific aspects, features and subsystem areas of the present invention will be referred to in greater detail below. In the drawings, like reference numerals denote like or corresponding parts throughout the various drawings and figures.

## Detailed Description of the Invention for Memory Addressing

In accordance with the present invention, a method and apparatus for addressing memory is described herein. In particular, the present invention provides for deferring variable width bit fields with fixed width words. More particularly, the present invention provides a method of addressing variable width data with a fixed width word. In various forms of the embodiment, variable bit field is used to specify bits to be substituted into the word or to specify an unused portion of the word in addressing variable width data with a fixed width word. In addition, the system of the present invention includes a microcodable state machine having an arithmetic core.

The microcodable state machine is intended to be used for solving design problems where there is a need for versatile and/or complicated calculations. Examples of such designs include address generation, stream parsing and decoding, and filter tap coefficient calculations. In this regard, the addressing must cope with two different features: (1) variable length addresses to access varying width portions of words and (2) address substitution. In the present invention, a RAM having a 64 x 32 bit configuration can



be addressed in partial words having 64 x 32 bit, 128 x 16 bit, 256 x 8 bit, 512 x 4 bit, 1024 x 2 bit, or 2048 x 1 bit formats.

#### Variable Length Fields Within a Fixed Width Word

In many applications, it is useful to define variable portions of a word (to be known as fields) for actions such as substitution, variable width data addressing, or the constriction of other parts of the word. The conventional method for defining variable portions of words is to have an additional word (or words) which specify the width of the field (or fields) within the word. In accordance with the present invention, a method for encoding this information within the word itself is described. The present method has the advantages of savings bits in the overall definition of the word, simplifying decoding of the encoded word and providing a more intuitive view of what has been encoded. Furthermore, this encoding method is applicable if the variable width fields are most or least significant bit justified within the word.

Accordingly, Table 1 shows two examples of variable width fields (marked "F") that are least significant bit justified defined within an eight bit word. A "w" marks other potential fields of these words.

Table 1

Bit number (hex)	7	6	5	4	3	2	1	0
Fixed word	w	w	w	F	F	F	F	F
	w	w	w	w	w	w	F	F

Table 2 shows the conventional method of encoding the fields shown in Table 1 using sufficient additional bits to specify the maximum width of the field in binary. (Bits marked "x" are "don't care", i.e., their value is of no consequence. This method is clearly inefficient in its use of bits and, furthermore, provides a less intuitive form than that described in the present invention.

Table 2

Bit number (hex)	7	6	5	4	3	2	1	0	Field Define	
Fixed word	w	w	w	x	x	x	x	x	1	0
	w	w	w	w	w	w	x	x	0	1

The new method, in accordance with the present invention, defines the field within the word. This method defines the field by using a continuation marker and a termination marker. The field is specified, from one end of the field, as

a series of continuation markers followed by a termination marker. In the case of a zero length field, however, only a termination marker is provided at the end of the word. Both the continuation marker and the termination marker are single bits, and they must be complementary. In addition, the field must be justified to either end of the word. Accordingly, the method of the present invention for encoding fields requires a width of only one bit extra over the original word width.

As shown in Table 3, the encoding of the fields shown in the Table 1, in accordance with the new method, is depicted. In this example, the continuation marker is "1" and the termination marker is "0". The field in this example is least significant bit justified.

Table 3

Bit number (hex)	7	6	5	4	3	2	1	0	
Fixed word	w	w	w	0	1	1	1	1	1
Continuation marker = 1;									
Termination marker = 0.	w	w	w	w	w	0	1	1	

Therefore, the advantages of the encoding method, in accordance with the present invention, are:

1. A reduction in the number of bits needed in the encoding.
2. A simplification in the decoding process is required since the need for a "x to 1 of " decode of the "field define" shown in Table 1-2 that would normally be required is inherent in the encoding which is already in the form of 1 of  $2^2$ ; and
3. The encoding is in a more intuitive form allowing the field defined to be more easily identified.

Furthermore, the use of this encoding method of the present invention can also be used such that the termination marker and the continuation marker are inverted to provide that the encoding of Table 3 resembles that of Table 4. Hence, the use of "1" or "0" is used interchangeably throughout this application.

Table 4

Bit number (hex)	7	6	5	4	3	2	1	0	
Fixed word	w	w	w	1	0	0	0	0	0
Continuation marker = 1;									
Termination marker = 0.	w	w	w	w	w	1	0	0	

As previously identified, the field encoded must be justified to either end of the word. Table 5 illustrates most significant justified fields, i.e., these are encoded in a similar way to least significant bit justified fields except that the field reaches from the most significant bit (hereinafter MSB) towards the least significant bit (hereinafter "LSB") up to and including the first termination marker. The encoding of the fields shown in Table 5 are shown in Table 6.

Table 5

Bit number (hex)	7	6	5	4	3	2	1	0
Fixed word	F	F	F	F	F	w	w	w
	F	F	w	w	w	w	w	w

Table 6

Bit number (hex)	7	6	5	4	3	2	1	0
Fixed word	1	1	1	1	1	0	w	w
Continuation marker = 1;								
Termination marker = 0.	1	1	0	w	w	w	w	w

Moreover, fields may be encoded from the least significant and most significant ends of the word simultaneously. For example, the two fields shown in Table 7 may be encoded as in Table 8, with the addition of just one bit for each field as described previously.

Table 7

Bit number (hex)	7	6	5	4	3	2	1	0
Fixed word	F	F	F	F	w	w	F	F
	w	w	w	w	F	F	F	F

Table 8

Bit number (hex)		7	6	5	4	3	2	1	0
Faced word	1	1	1	1	0	w	w	0	1
Continuation marker = 1;									
Termination marker = 0.	0	w	w	w	w	0	1	1	1

### Using a Fixed Width Word with Variable Length Fields to Perform Address Substitution

There are situations in which it is useful to substitute part of a memory address by another value. In this way it is possible to construct a data dependent address. The encoding method of the present invention can be applied to the addresses of a memory to specify what portion of the address is to be substituted. If a least significant bit justified variable length field is used in the address, a substitution field can be defined. For example, a 12 bit address 0b000000000000 encoded to have its five least significant bit substituted by the 12 bit value 0bcccccccccc would be 0b0000000011111 and produce the address 0b000000cccccc. Table 9 shows the encoding for substitution into a 12 bit address.

Table 9: Address substitution

[illegible]

Addressing Variable Width Data with a Fixed Width Word

One embodiment of the present invention is for addressing a memory which can be accessed at its full width or in  $2^n$  widths up to its full width (these smaller words are called partial words). Hence, it will be shown how the variable field encoding of the present invention can be used to address this memory and to index those addresses into the memory.

To access a 64 x 32 bit Register file in widths of 32, 16, 8, 4, 2 and 1 bit requires different lengths of address, i.e., the implementation of this embodiment is a 64 x 32 bit memory which can be accessed as 64 x 32 bits, 128 x 16 bits, 256 x 8 bits, 512 x 4 bits, 1024 x 2 bits, or 2048 x 1 bit. It is seen that 5 bits are required to address one of the 64 x 32 bit locations, while 12 bits are required to address one of the 2048 x 1 bit locations. Hence, the addresses can be of variable length and, in fact, the width of the address specifies the address format of the memory. Accordingly, the address can be defined within a fixed word width by using a most significant justified variable width field which constricts the address and defines its width. This is illustrated in Table 10.

Table 10: Variable width addressing

Data Width	A	8	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	0	0
16	0	0	0	0	1	0	0	0	0	0	0
32	0	0	0	0	0	1	0	0	0	0	0

To allow indexing of the address, a portion of it can be substituted using the same method described previously for address substitution. The substitution portion (or field) of the address can be defined by a least significant bit justified variable length field (The continuation marker "1", termination marker "0" that is superimposed on top of those shown in Table 10. Using an address of an eight bit word, as an example, Table 11 shows how to define the number of the least significant bits to be substituted. The least significant bit added is the substitution indicator (marked "v"). The general case of a Fixed width word for substitution is shown in Figure 2.

Table 11: Address substitution

Bits to be substituted	A	9	8	7	6	5	4	3	2	1	0	w
0	0	0	0	1	x	x	x	x	x	x	x	0
1	0	0	0	1	x	x	x	x	x	x	0	1
2	0	0	0	1	x	x	x	x	x	0	1	1
3	0	0	0	1	x	x	x	x	0	1	1	1
4	0	0	0	1	x	x	x	0	1	1	1	1
5	0	0	0	1	x	x	0	1	1	1	1	1
6	0	0	0	1	x	0	1	1	1	1	1	1
7	0	0	0	1	x	0	1	1	1	1	1	1
8	0	0	0	1	0	1	1	1	1	1	1	1

In effect the substitute code is superimposed on top of the address that is already coded. From this coding, it can be seen that there are illegal addresses, most obviously 0x0000 and 0x3fff. In this case, a "0" must be in the bottom 9 bits to prevent substituting more than 8 bits and a "1" in the top 8 bits specifies an allowable access width. If one of these errors is detected, the access is undefined, but the Register file contents will not be affected.

In accordance with the present invention, the system for addressing and for accessing partial words in a register file is discussed below.

The conventional memory circuitry dictates that the memory must always be accessed at its full width. To achieve variable width accesses, a full (32 bit) width word is read. This full word is rotated until the partial word accessed is justified in the LSB. The upper parts of the word are extended to the full width and then output. Extending may encompass padding with zeros or ones, sign extending, using the sign bit of a sign-magnitude number as the new MSB or any similar conventional method. Extending is dependent on the mode of operation. When the partial word is input to and written back into the memory, it is multiplexed back into the rotated full word, which is then rotated back and written into the array. Figure 3 shows these steps for the access of a 4 bit partial word in the fourth four bit word of the 32 bit word.

To access or read partial words, such as the highlighted four bit word shown in row "1" 213 of Figure 3, the full width word must be rotated to place the partial word at the LSB, as shown in row "2" 214. As shown in row "3" 215, the four bit word is extended to create a full 32 bit word. This word can now be accessed.

27

As shown in figure 3, a full width word that has been selected to be written back is truncated to the width of the original partial word which is multiplexed into the word shown in row "2" 214. At the LSB position, this is shown in row "4" 216. The resulting word is rotated back in its original significance in the read word, this is shown in row "5" 217. This full word can now be written back into the register file.

The following list, therefore, summarizes the steps numbered in Figure

3:

1. Full word read from memory;
2. 12 bit rotated right puts partial word into the LSB;
3. Extended to full word, then passed to output;
4. The inputted partial word is multiplexed into rotated full word from (2);
- and
5. 12 bit rotated left puts full word back to original state to be written.

The above accesses suggests the data flow structure of the memory that is shown in Figure 4. The numbers in the structure refer to the above text and to Figure 3.

The memory address must be decoded to control the above structure. It should be recognized that the MSB of any width of address is at the same significance with reference to the memory. The top six bits of a decoded address are a 32 bit word address, whereas the remainder is a bit address. Therefore, the stage of decoding (in parallel with the substitution) is to decode the address width defining variable field by detecting the position of the most significant termination marker. This allows the address to be MSB justified (shifting in zeros at the LSB). The top six bits can be used directly as a 32 bit word row address of the memory. The bottom five bits can be used to directly control both barrel shifters (as seen in Figure 4), because, for example, an original 32 bit address will always have a shift of 0b00000 (these having been shifted when the address was MSB justified). Similarly, a 16 bit address can have a shift of 0bx0000, i.e., 0 or 16 bit shift and a 1 bit address can have a shift of 0bxxxxx, i.e., 0 to 31 bit shifts. The extender and input multiplexer are controlled by the access width decode to mask out the output words and multiplex the input words to an appropriate significance, respectively. The block diagram of the decode is shown in Figure 5. It can be seen that the decode of the two variable width fields for width and substitution can be done in parallel and independently.

Figure 2 illustrates an example of a fixed width word 13 bits long for addressing variable width data and substitution as shown in the bottom two rows. For these examples, an eight bit word would have been addressed at location 0b110ssss, where "ssss" is substituted from another address source.

#### 5 Microcodable State Machine Structure

In accordance with the present invention, the substitution into a memory address and the variable width accessing of a memory have been brought together in the implementation of a microcodable state machine the structure of which is shown in Figure 6. The structure is one of a state machine 218 providing control of an arithmetic core 219 by way of a wide word of control signals called a microcode instruction. The arithmetic core 219, in turn, passes status flags and some data to the state machine 218.

The state machine 218, in accordance with the present invention, includes a memory containing a list of the microcode instructions. As with conventional microcodable state machines, it is capable of either proceeding through the list of microcode instructions contiguously or a jump can occur from one instruction to another. The jump address is in the form shown in Figure 7. The substituted value comes from the Arithmetic core 219 as shown in Figures 6 and 8. This allows the construction of "jump tables" within the microcode programs. Thus, if a jump is made with 3 bits substituted, for example, there are eight possible contiguous locations that may be jumped to, each dependent on the value from the arithmetic core, i.e., it has so become a programmable jump.

#### 20 Arithmetic Core

25 The arithmetic core 219, as shown in Figure 8, includes a memory called a register file 221, an Arithmetic and Logic unit (ALU) 222, an input port 223 and an output port 224. These components are connected via buses and multiplexers. As previously stated, these components, and the multiplexers defining their connections, are entirely controlled by the microcode instruction issued by the state machine 218. The ALU 222 and the ports 223 and 224 are conventional, however, the register file 221 is a memory which allows variable width indexed accesses. The addresses to the register file 221 is coded directly into the microcode instruction.

35 There are many advantages of using this method of addressing to the register file. First, many locations in an application do not need to be the full width of the memory (32 bits in this case). Whilst it will cause no effect on the operation of the device to use a full width location, it is very wasteful of



memory locations. Minimizing the number of memory locations will minimize the amount of space used by the memory and, therefore, minimize the capacitive loading in the register file. This maximizes the speed of the register file. Second, the indexing combined with the variable width of memory accessing allows the stepping through of locations of variable width. In the one bit case this allows an elegant implementation of long division and multiplication.

In summary, therefore, there is described a procedure for addressing memory having the following steps: (1) providing a fixed width word having a predetermined fixed number of bits to be used for addressing variable width data; (2) defining the fixed width word with a width defining field and an address field providing the width defining field with at least one bit to serve as a termination marker; (3) defining the address field with a plurality of bits defining the address of the data; and (4) varying the size of bits in the address field in inverse relation to the size of the variable width data varying the number of bits in the width defining field in direct relation to the size of the variable width data and maintaining a fixed width word for addressing variable width data while varying the width of the width defining field and the address field. In addition, a procedure for addressing memory having the following steps is described: (1) providing a fixed width word having a predetermined fixed number of bits to be used for addressing data; (2) defining the fixed width word with an address field and a substitution field; (3) defining the address field with a plurality of bits defining the address of the data; (4) defining a variable width substitution field with at least one substitution bit; (5) the substitution field has at least one bit to serve as a termination marker between the address field and the substitution field; and (6) using the substitution field to indicate substituted bits from a separate addressing source and maintaining a fixed width word for addressing variable width data while inversely varying the width of the address field and the width of the substitution field. In addition, a process for addressing variable width data in a memory is described as having the following steps: (1) providing a memory having words of predetermined width and composed of partial words; (2) rotating the partial word to be accessed to a least significant bit justification; (3) extending the remaining part of the word so that the accessed word will be recognized as the partial word; and (4) restoring the remaining part of the word and rotating the word until the partial word is restored to its original position.

**Detailed Description of the Invention for Transforming Data Using a Common Processing Block**

This present embodiment, in accordance with the present invention, relates to a method for the transformation of signals from a frequency to a time representation, as well as a digital circuit arrangement for implementing the transformation.

It is a common goal in the area of telecommunications to increase both information content and transmission speed. Each communications medium, however, imposes a limitation on transmission speed, as does the hardware at the transmitting and receiving end that must process the transmitted signals. A telegraph wire is, for example, typically a much faster medium for transmitting information than the mail is, even though it might be faster to type and read a mailed document than to tap out a telegraph key.

The method of encoding transmitted information also limits the speed at which information can be conveyed. A long-winded telegraph message will, for example, take longer to convey than a succinct message with the same information content. The greatest transmission and reception speed can therefore be obtained by compressing the data to be transmitted as much as possible, and then, using a high-speed transmission medium, to process the data at both ends as fast as possible, which often means the reduction or elimination of 'bottlenecks' in the system.

One application in which it is essential to provide high-speed transmission of large amounts of data is in the field of digital television. Whereas conventional television systems use analog radio and electrical signals to control the luminance and color of picture elements ('pixels') in lines displayed on a television screen, a digital television transmission system generates a digital representation of an image by conveying analog signals into binary 'numbers' corresponding to luminance and color values for the pixels. Modern digital encoding schemes and hardware structures typically enable much higher information transmission rates than do conventional analog transmission systems. As such, digital televisions are able to achieve much higher resolution and much more life-like images than their conventional analog counterparts. It is anticipated that digital television systems including so-called High-Definition TV (HDTV) systems, will replace conventional analog television technology within the next decade in much of the industrialized world. The conversion from analog to digital imaging, for both transmission

and storage will, thus, be similar to the change-over from analog audio records to the now ubiquitous compact discs (CD's).

In order to increase the general usefulness of digital image technology, standardized schemes for encoding digital images have been adopted. Once such standardized scheme is known as the JPEG standard and is used for still pictures. For moving pictures, there are at present two standards, MPEG and H.261, both of which carry out JPEG-like procedures on each of the sequential frames of the moving picture. To gain advantage over using JPEG repeatedly, MPEG and H.261 operate on the differences between subsequent frames, taking advantage of the well-known fact that the difference, that is, the movement between frames, is small. It, therefore, takes less time or space to transmit or store the information corresponding to the changes rather than to transmit or store equivalent still-picture information as if each frame in the sequence were completely unlike the frames closest to it in the sequence.

For convenience, all the current standards operate by breaking an image or picture into files or blocks, each block consisting of a piece of the picture eight pixels wide by eight pixels high. Each pixel is then represented by three (or more) digital numbers known as 'components' of that pixel. There are many different ways of breaking a colored pixel into components, for example, using standard notation, e.g., YUV, YCr, Cb, RGB, etc. All the conventional JPEG-like methods operate on each component separately.

It is well known that the eye is insensitive to high-frequency components (or edges) in a picture. Information concerning the highest frequencies can usually be omitted altogether without the human viewer noticing any significant reduction in image quality. In order to achieve this ability to reduce the information content in a picture by eliminating high-frequency information without the eye detecting any loss of information, the 8-by-8 pixel block containing spatial information (for example, the actual values for luminance) must be transformed in some manner to obtain frequency information. The JPEG, MPEG and H.261 standards all use the known Discrete Cosine Transform to operate on the 8-by-8 spatial matrix to obtain an 8-by-8 frequency matrix.

As described above, the input data represents a square area of the picture. In transforming the input data into the frequency representation, the transform that is applied must be two-dimensional, but such two-dimensional transforms are difficult to compute efficiently. The known, two-dimensional Discrete Cosine Transform (DCT) and the associated inverse DCT (IDCT),

however, have the property of being "separable". This means that rather than having to operate on all 64 pixels in the eight-by-eight pixel block at one time, the block can first be transformed row-by-row into intermediate values, which are then transformed column-by-column into the final transformed frequency values.

A one-dimensional DCT of order N is mathematically equivalent to multiplying two N-by-N matrices. In order to perform the necessary matrix multiplication for an eight-by-eight pixel block, 512 multiplications and 448 additions are required, so that 1,024 multiplications and 896 additions are needed to perform the full 2 dimensional DCT on the 8-by-8 pixel block. These arithmetic operations, and especially multiplication, are complex and slow and, therefore, limit the achievable transmission rate. They also require considerable space on the silicon chip used to implement the DCT.

The DCT procedure can be rearranged to reduce the amount of computation required. There are, at present, two main methods used for reducing the computation required for the DCT, both of which use "binary decimation". The term "binary decimation" means that an N-by-N transform can be computed by using two N/2-by-N/2 transformations, plus some computational overhead whilst arranging this. Whereas the eight-by-eight transform requires 512 multiplications and 448 additions, a four-by-four transform requires only 64 multiplications and 48 additions. Binary decimation, thus, saves 284 multiplications and 352 additions and the overhead incurred in performing the decimation is typically insignificant compared to the reduction in computation.

At present, the two main methods for binary decimation were developed by Eong Gi Lee ('A New Algorithm to Compute the DCT') IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. Assp 32, No 6, p 1243 December 1984) and Wen-Hsiung Chen ('A Fast Computational Algorithm for the DCT', Wen-Hsiung Chen, C. Harrison Smith, S C Pralick, IEEE Transactions on Communications, Col. Com 25, No. 9 1004, September 1977). Lee's method makes use of the symmetry inherent in the definition of the inverse DCT and, by using simple cosine identities, it defines a method for recursive binary decimation. The Lee approach is only suitable for the IDCT.

The Chen method uses a recursive matrix identity that reduces the matrices into diagonals only. This method provides easy binary decimation of the DCT using known identities for diagonal matrices.

A serious disadvantage of the Lee and Chen methods is that they are unbalanced in respect of when multiplications and additions must be performed. Essentially, both of these methods require that many additions be followed by many multiplications, or vice versa. When implementing the Lee or Chen methods in hardware, it is, therefore, not possible to have parallel operation of adders and multipliers. This reduces their speed and efficiency since the best utilization of hardware is when all adders and multipliers are used all the time.

An additional disadvantage of such known methods and devices for performing DCT and IDCT operations is that it is usually difficult to handle the so-called normalization coefficient, and known architectures require adding an additional multiplication time when all the multipliers are being used.

Certain known methods for applying the forward and Inverse DCT to video data are very simple and highly efficient for a software designer who need not be concerned with the layout of the semiconductor devices which perform the calculations. Such methods, however, often are far too slow or are too complex in semiconductor architecture and hardware interconnections to perform satisfactorily at the transmission rate desired for digital video.

Yet another shortcoming of existing methods and hardware structures for performing DCT and IDCT operations on video data is that they require floating-point internal representation of numerical values. To illustrate this disadvantage, assume that one has a calculator that is only able to deal with three - digit numbers, including digits to the right of the decimal point (if any). Assume further that the calculator is to add the numbers 12.3 and 4.58 (Notice that the decimal point is not fixed relative to the position of the digits in these two numbers. In other words, the decimal point is allowed to "float"). Since the calculator is not able to store the four digits required to fully represent the answer 16.88, the calculator must reduce the answer to three digits either by truncating the answer by dropping the right-most "8", yielding an answer of 16.8, or it must have the necessary hardware to round the answer up to the closest three-digit approximation 16.9.

As this very simple example illustrates, if floating-point arithmetic is required, one must either accept a loss of precision or include highly complicated and space-wasting circuitry to minimize rounding error. Even with efficient rounding circuitry, however, the accumulation and propagation of rounding or truncation errors may lead to unacceptable distortion in the video signals. This problem is even greater when the methods for processing the

video signals require several multiplications, since floating point rounding and truncation errors are typically greater for multiplication than for addition.

A much more efficient DCT/IDCT method and hardware structure would ensure that the numbers used in the method could be represented with a fixed decimal point, but in such a way that the full dynamic range of each number could be used. In such a system, truncation and rounding errors would either be eliminated or, at least, greatly reduced.

In the above example, if the hardware can handle four digits, no number greater than 99.99 were ever needed, and every number had the decimal point between the second and third places, then the presence of the decimal point would not affect calculations at all. Accordingly, the arithmetic could be carried out just as if every number were an integer, e.g., the answer  $1230+0456=1686$  would be just as clear as  $12.30+4.56=16.86$ , since one would always know that the '1686' should have a decimal point between the middle '6' and '8'. Alternatively, if numbers (constant or otherwise) are selectively scaled or adjusted so that they all fall within the same range, each number in the range could also be accurately and unambiguously represented as a set of integers.

One way of reducing the number of multipliers needed is simply to have a single multiplier that is able to accept input data from different sources. In other words, certain architectures use a single multiplier to perform the multiplications required in different steps of the DCT or IDCT calculations. Although such "crossbar switching" may reduce the number of multipliers required, it means that large complicated multiplexer structures must be included instead to select the inputs to the multiplier, to isolate others from the multiplier, and to switch the appropriate signals from the selected sources to the inputs of the multiplier. Additional large-scale multiplexers are also required to switch the large number of outputs from the shared multipliers to the appropriate subsequent circuitry. Crossbar switching or multiplexing is, therefore, complex, is generally slow (because of the extra storage needed) and costs are significant in a final semiconductor implementation.

Still another drawback of existing architectures, including the "crossbar switching" is that they require general purpose multipliers. In other words, existing systems require multipliers for which both inputs are variable. As is well known, implementations of digital multipliers typically include rows of adders and shifters such that, if the current bit of a multiplier word is a 'one' the value of the multiplicand is added into the partial result, but not if the

current bit is a 'zero'. Since a general purpose multiplier must be able to deal with the case in which every bit is a '1', a row of adders must be provided for every bit of the multiplier word.

By way of example, assume that data words are 8 bits wide and that one wishes to multiply single inputs by 5. An 9-bit representation of the number 5 is 00000101. In other words, digital multiplication by 5 requires only that the input value be shifted to the left two places (corresponding to multiplication by 4) and then added to its up-shifted value. The other six positions of the coefficients have bit values of '0', so they would not require any shifting or additional steps.

A fixed-coefficient multiplier, that is, in this case, a multiplier capable of multiplying only by five, would require only a single shifter and a single adder in order to perform the multiplication (disregarding circuitry needed to handle carry bits). A general purpose multiplier, in contrast, would require shifters and adders for each of the eight positions, even though six of them would never need to be used. As the example illustrates, fixed coefficients can simplify the multipliers since they allow the designer to eliminate rows of adders that correspond to zeros in the coefficient, thus saving silicon area.

In an IDCT method, in accordance with the present invention, a one-dimensional IDCT for each N-row and N-column of N-by-N pixel blocks is decimated and a 1-D IDCT is performed separately on the N-2 even-numbered pixel input words and the N-2 odd-numbered pixel input words.

In a preferred embodiment, N=8 according to the JPEG standard. The two-dimensional IDCT result is then obtained by performing two one-dimensional IDCT operations in sequence (with an intermediate reordering-transposition-of data).

In a common processing step, for N=8, a first pair of input values is passed without need for multiplication to output adders and subtractors. Each of a second pair of input values is multiplied by each of two constant-coefficient values corresponding to two scaled cosine values. No other multiplications and only one subtraction and one addition are required in the common processing step. The second pair is then added or differenced pairwise with the first pair of input values to form even or odd resultant values.

In a pre-common processing stage, the lowest order odd input word is pre-multiplied by the square root of two and the odd input words are summed pairwise before processing in the common processing block. In a post-common processing stage, intermediate values corresponding to the pro-

cessed odd input words are multiplied by predetermined constant coefficients to form odd resultant values.

After calculation of the even and odd resultant values, the N/2 high-order outputs are formed by simple subtraction of the odd resultant values from the even resultant values, and the N/2 low-order outputs are formed by simple addition of the odd resultant values and the even resultant values.

For both the DCT (at the transmission end of a video processing system) and the IDCT (at the receiving end, which incorporates one or more of the various aspects of the present invention), the values are preferably and deliberately scaled downward by a factor of two by a simple binary right shift. This deliberate, balanced, upward scaling eliminates several multiplication steps that are required according to conventional methods.

According to another aspect of the method, in accordance with the present invention, selected bits of constant coefficient or intermediate resulting data words are rounded or adjusted by predetermined setting of selected bits to either '1' or '0'.

Two-dimensional transformation of pixel data is carried out by a second, identical 1-D operation on the output values from the first 1-D IDCT processing steps.

An IDCT system, according to yet another aspect of the present invention, includes a pre-common processing circuit, and a common processing circuit, in which the pre-common, common, and post-common processing calculations are performed on input data words. A supervisory controller generates control signals to control the loading of various system latches; preferably, to serially time-multiplex the application of the N/2 even and N/2 odd-numbered input words to input latches of the pre-common block to direct addition of the even and odd resultant values to form and latch low order output signals and to direct subtraction of the odd resultant values from the even resultant values to form and latch the high-order output signals and to sequentially control internal multiplexers.

In the present invention, even and odd input words are preferably processed in separate passes through the same processing blocks. Input data words are preferably (but not necessarily) latched, not in strictly ascending or descending order, but rather in an order enabling an efficient 'butterfly' structure for the data path.

Furthermore, at least the common processing circuit may be configured as a pre-logic circuit, with no clock or control signals required for its



proper operation, as may be other processing blocks, depending on the particular application.

No general-purpose multipliers (with two variable inputs) are required. Rather, constant coefficient multipliers are included throughout the preferred embodiment. Furthermore, fixed-point integer arithmetic devices are included in the preferred embodiment of the invention and can be so designed as to provide a method and system for performing IDCT transformation of video data with one or more of the following features:

1. Constant use of all costly arithmetic operations;
2. In order to reduce the silicon area needed to implement the IDCT, there are a small number of storage elements (such as latches), preferably no more than required for efficient pipelining of the architecture, coupled with a small number of constant coefficient multipliers rather than general purpose multipliers that require extra storage elements;
3. Operations are arranged so that each arithmetic operation does not need to use sophisticated designs, for example, if known 'ripple adders' are used, these would allow sufficient time to 'resolve' (see below) or produce their answers; if operations are arranged in such a way that other devices precede the rearranging operations so as to avoid delay and to allow greater throughput and efficiency;
4. One is able to generate results in a natural order;
5. No costly, complex, crossbar switching is required;
6. The architecture is able to support much faster operations; and
7. The circuitry used to control the flow of data through the transform hardware can be small in area.

#### Theoretical Background of the Invention

In order to understand the purpose and function of the various components and the advantages of the signal processing method used in the IDCT system according to the present invention, it is helpful to understand the system's theoretical basis.

#### Separability of a Two-Dimensional IDCT

The mathematical definition of a two-dimensional forward discrete cosine transforms (DCT) for an  $N \times N$  block of pixels is as follows, where  $U(j,k)$  are the pixel frequency values corresponding to the pixel absolute values  $X(m,n)$

Equation 1:

$$Y(j,k) = \frac{2}{N} c(j) c(k) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} X(m,n) \cos\left[\frac{(2m-1)jn}{2N}\right] \cos\left[\frac{(2n-1)kn}{2N}\right]$$

where  $j, k = 0, 1, \dots, N-1$  and

$$c(j), c(k) = 1/\sqrt{2} \quad \text{for } j, k = 0; \text{ otherwise } 1$$

5 The terms  $2N$  govern the dc level of the transform, and the coefficients  $c(j)$ ,  $c(k)$  are known normalization factors.

The expression for the corresponding inverse discrete cosine transform, that is for the IDCT, is as follows:

Equation 2:

$$X(m,n) = \frac{2}{N} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} c(j) c(k) Y(j,k) \cos\left[\frac{(2m-1)jn}{2N}\right] \cos\left[\frac{(2n-1)kn}{2N}\right]$$

10

where  $j, k = 0, \dots, N-1$  and

$$15 \quad c(j), c(k) = 1/\sqrt{2} \quad \text{for } j, k = 0; \text{ otherwise } 1$$

The forward DCT is used to transform spatial values (whether representing characteristics such as luminance directly, or representing differences, such as in the MPEG standard) into their frequency representation. The inverse DCT, as its name implies, operates the other 'direction', that is, the IDCT transforms the frequency values back into spatial values.

20 In the expression, Equation 2, (E2), note that the cosine functions each depend on only one of the summation indices.

The expression E2 can therefore be rewritten as:

Equation 3:

$$x(m,n) = \frac{2}{N} \sum_{j=0}^{N-1} c(j) \cos \left[ \frac{(2m-1)jn}{2N} \right] \sum_{k=0}^{N-1} c(k) y(j,k) \cos \left[ \frac{(2n-1)km}{2N} \right]$$

This is the equivalent of a first one-dimensional IDCT performed on the product of all terms that depend on k and n, followed, after a straightforward standard data transposition by a second one-dimensional IDCT using as

5 inputs the outputs of the first IDCT operation.

#### Definition of the 1-D IDCT

A 1-dimensional N-point IDCT (where n is an even number) is defined by the following expression.

Equation 4:

$$X(k) = \sum_{n=0}^{N-1} c(n) \cdot y(n) \cos \left[ \frac{n(2k-1)}{2N} \right] \quad k = \{0, 1, \dots, N-1\}$$

$$c(n) = 1/\sqrt{2} \quad \text{for } n=0; \text{ otherwise } 1$$

10 and where y(n) are the N inputs to the inverse transformation function and x(k) are its N outputs. As in the 2-D case, the formula for the DCT has the same structure under the summation sign, but with the normalization constant outside the summation sign and with the x and y vectors switching places in the equation.

#### Resolution of a 1-D IDCT

As is shown above, the 2-D IDCT can be calculated using a sequence of 1-D IDCT operations separated by a transpose. In accordance to one embodiment, each of these 1-D operations is, in turn, broken down into sub-procedures that are then exploited to reduce even further the required size and complexity of the semiconductor implementation.

20

#### Normalization of Coefficients

As is discussed above, an important design goal for IDCT hardware is the reduction of the required number of multipliers that must be included in the circuitry. Most methods for calculating the DCT of IDCT, therefore, attempt to reduce the number of multiplications needed. According to this embodiment, however, all the input values are deliberately scaled upward by a factor of the

25

square root of two. In other words, using the method according to this embodiment of the present invention, the right-hand side of the IDCT expression (E) is deliberately multiplied by the square root of two.

According to this embodiment, two 1-D IDCT operations are performed in series (with an intermediate transpose) to yield the final 2-D IDCT result. Each of these 1-D operations includes a multiplication by the same square root of two factor. Since the intermediate transposition involves no scaling, the result of two multiplications by the square root of two in series is that the final 2-D results will be scaled upward by a factor two. To obtain the unscaled value, the circuitry need then only divide by two. Since the values are all represented digitally, this can be accomplished easily by a simple right shift of the data. As is made clearer below, the upward scaling by the square root of two in each 1-D IDCT stage and final down-scaling by 2 is accomplished by adders, multipliers and shifters all within the system's hardware, so that the system places no requirements for scaled inputs on the other devices to which the system may be connected. Because of this, the system is compatible with other conventional devices that operate according to the JPEG or MPEG standards. Normalization according to this embodiment of the present invention, therefore, eliminates the need for hardware multipliers within the IDCT semiconductor architecture for at least two square root of two multiplication operations. As is explained below in greater detail, the single additional multiplication step (upward scaling by the square root of two) of the input data in each 1-D operation leads to the elimination of still other multiplication steps that are required when using conventional methods.

#### 25 Separation of the 1-D IDCT into High and Low-Order Outputs

Expression E can now be evaluated separately for the  $N/2$  low-order outputs ( $k=0, 1, \dots, N/2-1$ ) and the  $N/2$  high order outputs ( $k=N/2, k=N/2+1, \dots, N$ ). For  $N=8$ , this means that one can first transform the inputs to calculate  $y(0)$ ,  $y(1)$ ,  $y(2)$  and  $y(3)$ , and then transform the inputs to calculate  $y(4)$ ,  $y(5)$ ,  $y(6)$  and  $y(7)$ .

Introduce the variable  $k'=(N-1-k)$  for the high-order outputs ( $k=N/2+1, \dots, N$ ), so that  $k'$  varies from  $(N/2-1)$  to  $N$  as  $k$  varies from  $(N/2+1)$  to  $N$ . For  $N=8$ , this means that  $k'=(3,2,1,0)$  for  $k=(4,5,6,7)$ . It can then be shown that expression E can be divided into the following two subexpressions E5 (which is the same as E except for the interval of summation) and E6:

Low order outputs:

Equation 5:

$$x(k) = \sum_{n=0}^{N/2-1} c(n) \cdot y(n) \cdot \cos\left[\frac{n(2k+1)\pi}{2N}\right]$$

where  $k=\{0,1,\dots,(N/2-1)\}$ ; and

$$c(n) = 1/\sqrt{2} \quad \text{for } n \text{ odd; otherwise } 1$$

5 High-order outputs:

Equation 6:

$$x(k) = x(N-1-k) = \sum_{n=0}^{N/2-1} y(n) (-1)^n \cos\left[\frac{n(2k+1)\pi}{2N}\right]$$

where  $k=\{N,\dots,(N/2+1)\} \rightarrow k'=\{0,1,\dots,(N/2-1)\}$

(Since  $c(n)=1$  for all high-order terms,  $c(n)$  is not included in this expression)

- 10 Note that both E5 and E6 have the same structure under the summation sign except that the term  $(-1)^n$  changes the sign of the product under the summation sign for the odd-numbered inputs ( $n$  odd) for the upper  $N/2$  output values and except that the  $y$  term will be multiplied by  $c(O) = 1/\sqrt{2}$ .

Separation of the 1-D IDCT into Even and Odd Inputs

- 15 Observe that the single sum in the 1-D IDCT expression E4 can also be separated into two sums: one for the even-numbered inputs (for  $N=8$   $y(0)$ ,  $y(2)$ ,  $y(4)$ , and  $y(6)$ ) and one for the odd-numbered inputs (for  $n=8$   $y(1)$ ,  $y(3)$ ,  $y(5)$ , and  $y(7)$ ). Let  $g(k)$  represent the partial sum for the even-numbered inputs and  $h(k)$  represent the partial sum for the odd-numbered inputs.

Thus:

Equation 7.

$$g(k) = \sum_{n=0}^{N/2-1} a(2n) y(2n) \cos\left[\frac{\pi(2k-1)2n}{2N}\right] + \sum_{n=0}^{N/2-1} c(2n) y(2n) \cos\left[\frac{\pi(2k-1)n}{2\left(\frac{N}{2}\right)}\right]$$

Where  $k=\{0,1,\dots,(N/2-1)\}$ ; and

Equation 8.

$$h(k) = \sum_{n=0}^{N/2-1} y(2n-1) \cos\left[\frac{\pi(2k-1)(2n-1)}{2N}\right]$$

5

where  $k=\{0,1,\dots,(N/2-1)\}$ .

For  $N=8$ , observe that the sums in E7 and E8 both are taken over

$n=\{0,1,2,3\}$ .

Now recall the known cosine identity:

$$2\cos A \cos B = \cos(A+B) + \cos(A-B),$$

10 and set  $A = \pi(2k-1)y(2N)$  and  $B = \pi(2k-1)(2N-1)/2N$ .

One can then multiply both sides of the expression E8 by:

$$2\cos A = 1/\{2\cos[\pi(2k-1)/2N]\} = C_k.$$

Note that, since  $C_k$  does not depend on the summation index  $n$ , it can be moved within the summation sign. Assume then by definition that  $y(-1)=0$ , and note that

15 the cosine function for the input  $y(7)$  is equal to zero. The expression for  $h(k)$

can then be rewritten in the following form:

Equation 9.

$$h(k) = \frac{1}{2\cos\left(\frac{\pi(2k-1)}{2N}\right)} \sum_{n=0}^{N/2-1} [y(2n-1) + y(2n-1)] \cos\left[\frac{\pi(2k-1)n}{2\left(\frac{N}{2}\right)}\right]$$

20

Where  $k=\{0,1,\dots,(N/2-1)\}$ .

Note that the inputs  $[y(2n+1)+y(2n-1)]$  imply that in calculating  $h(k)$ , the odd input terms are paired to form  $N/2$  paired inputs  $p(n)=y(2n+1)+y(2n-1)$ .

For  $N=8$  the values of  $p(n)$  are as follows:

n	p(n)
0	$y(-1) + Y(1) = Y(1) Y(-1) = 0$ by definition
1	$y(1) + y(3)$
2	$y(3) + y(5)$
3	$y(5) + y(7)$

Expression E9 for  $h(k)$  can then be represented by the following:  
Equation 10.

$$h(k) = C_k \sum_{n=0}^{\frac{N}{2}-1} p(n) \cos\left[\frac{\pi(2k-1)n}{2 \left(\frac{N}{2}\right)}\right]$$

10 Where  $k=(0,1,\dots,(N/2-1))$ .

Observe now that the cosine term under the summation sign is the same for both  $g(k)$  and  $h(k)$  and that both have the structure of a 1-D IDCT (compared with expression E5). The

result of the IDCT for the odd  $k$  terms, that is, for  $h(k)$ , however is multiplied by

15 the factor

$$C_k = 1/(2 \cdot \cos[\pi(2k-1)/2N]).$$

In other words,  $g(k)$  is an  $n/2$ -point IDCT operating on even inputs  $y(2n)$  and  $h(k)$  is an  $n/2$ -point IDCT operating on  $[y(2n+1)-y(2n-1)]$  where  $y(-1)=0$  by definition.

20 Now introduce the following identities:

$$\begin{aligned} & y(n) - y(n); \\ & c1 - \cos(\pi/8); \\ & c2 - \cos(2\pi/8) - \cos(\pi/4) - 1/\sqrt{2}; \\ & c3 - \cos(3\pi/8); \\ & d1 - 1/[2 \cdot \cos(\pi/16)]; \\ & d3 - 1/[2 \cdot \cos(3\pi/16)]; \\ & d5 - 1/[2 \cdot \cos(5\pi/16)]; \text{ and} \\ & d7 - 1/[2 \cdot \cos(7\pi/16)]. \end{aligned}$$

25

Further introduce scaled cosine coefficients as follows:

$$c1s = \sqrt{2} \cdot \cos(\pi/8);$$

$$c3s = \sqrt{2} \cdot \cos(3\pi/8);$$

Using the evenness ( $\cos(-\phi) = \cos(\phi)$ ) and periodicity

( $\cos(\phi + \pi) = -\cos(\phi)$ ) of the cosine function, expressions E7 and

5 E8 can then be expanded for  $N=8$  to yield (recall also (C)) is  $1/\sqrt{2}$ :

$$g(0) = 1/\sqrt{2} \cdot y_0 + y_2c1 + y_4c2 + y_6c3 + 1/\sqrt{2} \cdot (y_1 + y_2c1s + y_4 + y_6c3s)$$

$$g(1) = 1/\sqrt{2} \cdot y_0 + y_2c3 - y_4c2 + y_6c1 + 1/\sqrt{2} \cdot (y_0 + y_2c3s - y_4 - y_6c1s)$$

$$g(3) = 1/\sqrt{2} \cdot y_0 + y_2c1 - y_4c2 - y_6c3 + 1/\sqrt{2} \cdot (y_0 - y_2c1s + y_4 - y_6c3s)$$

10 and

$$h(0) = d1 \cdot \{y1 + (y1 + y3) c1 - (y3 + y5) c2 + (y5 + y7) c3\} =$$

$$d1/\sqrt{2} \cdot \{\sqrt{2} \cdot y1 + (y1 + y3) c1s - (y3 + y5) c2s - (y5 + y7) c3s\}$$

$$h(1) = d3 \cdot \{y1 + (y1 + y3) c3 - (y3 + y5) c2 + (y5 + y7) c1\} - d3/\sqrt{2} \cdot$$

$$\{\sqrt{2} \cdot y1 + (y1 + y3) c3s - (y3 + y5) c2s - (y5 + y7) c1s\}$$

15  $h(2) = d5 \cdot \{y1 + y3\} c3 - (y3 + y5) c2 + (y5 + y7) c1 = d5/\sqrt{2} \cdot$

$$\{\sqrt{2} \cdot y1 - (y1 + y3) c3s - (y3 + y5) c2s - (y5 + y7) c1s\}$$

$$h(3) = d7 \cdot \{y1 - (y1 + y3) c1 + (y3 + y5) c2 - (y5 + y7) c3\} - d7/\sqrt{2} \cdot$$

$$\{\sqrt{2} \cdot y1 - (y1 + y3) c1s + (y3 + y5) c2s - (y5 + y7) c3s\}$$

20 Now, recall that according to this embodiment of the present invention, all values are scaled upward by a factor of 2 for both the DCT and IDCT operations. In other words, according to the embodiment, both  $h(k)$  and  $g(k)$  are multiplied by this scaling factor. The  $g(k)$  and  $h(k)$  expressions, therefore, become:

Equation 11.  $g(0) = y0 + y2 + c1s + y4 + y6 + c3s$

$$g(1) = y0 + y2 + c3s - y4 - y6 + c3s$$

25  $g(2) = y0 + y2 + c3s - y4 + y6 + c1s$

and  $g(3) = y0 - y2 + c1s + y4 - y6 + c3s$

Equation 12.

$$h(0) = d1 [\sqrt{2} \cdot y1 - (y1 + y3) c1s - (y3 + y5) c2s - (y5 + y7) c3s]$$

$$h(1) = d3 [\sqrt{2} \cdot y1 - (y1 + y3) c3s - (y3 + y5) c2s - (y5 + y7) c1s]$$

$$h(2) = d5 [\sqrt{2} \cdot y1 - (y1 + y3) c3s - (y3 + y5) c2s - (y5 + y7) c1s]$$

$$h(3) = d7 [\sqrt{2} \cdot y1 - (y1 + y3) c1s - (y3 + y5) c2s - (y5 + y7) c3s]$$



45

Notice that since  $c2 = \cos(\pi/4) = 1/\sqrt{2}$ , multiplication by  $\sqrt{2}$  gives a scaled  $c2$  value=1. By scaling the expressions (corresponding to upward scaling of the values of the video absolute and frequency values) according to this embodiment, it is, therefore, possible to eliminate the need to multiply and  $c3s$ , both of which are constant coefficients so that general utility multipliers are not needed. This, in turn, eliminates the need for the corresponding hardware multiplier in the semiconductor implementation of the IDCT operations.

The similarity in structure of  $g(k)$  and  $h(k)$  can be illustrated by expressing these sets of equations in matrix form. Let  $C$  be the  $4 \times 4$  cosine coefficient matrix defined as follows:

Equation 13.

$$C = \begin{bmatrix} 1 & c1s & 1 & c3s \\ 1 & c3s & -1 & -c1s \\ 1 & -c3s & -1 & c1s \\ 1 & -c1s & 1 & -c3s \end{bmatrix}$$

Equation 14

$$\begin{bmatrix} g(0) \\ g(1) \\ g(2) \\ g(3) \end{bmatrix} = C \cdot \begin{bmatrix} y0 \\ y2 \\ y4 \\ y6 \end{bmatrix}$$

Equation 15.

$$\begin{bmatrix} h(0) \\ h(1) \\ h(2) \\ h(3) \end{bmatrix} = D \cdot C \cdot \begin{bmatrix} \sqrt{2} \cdot y1 \\ y1 + y3 \\ y3 + y5 \\ y5 + y7 \end{bmatrix}$$

15

Where  $D = \text{diag}[d1, d3, d5, d7]$ —the  $4 \times 4$  matrix with  $d1, d3, d5$ , and  $d7$  along the diagonal and with other elements equal to zero. As E14 and E15 show, the procedures for operating on even-numbered inputs to get  $g(k)$  and for operating

on the odd-numbered inputs to get  $h(k)$  both have the common step of multiplication by the cosine coefficient matrix  $C$ . To get  $h(k)$ , however, the inputs must first be pairwise summed (recalling that  $y(-1)=0$  by definition),  $y(1)$  must be premultiplied by 2, and the result of the multiplication by  $C$  must be multiplied by

5 D.

As the expressions above also indicate, the N-point, 1-D IDCT (see E4) can also be split into the two N/2-point, 1-D IDCT's each involving common core operations (under the summation sign) on the N/2 odd (grouped) and the N/2 even input values. The expressions above yield the following simple structure for

10 the IDCT as implemented in this embodiment:

Low-order outputs for (N=8, outputs  $k=\{0,1,2,3\}$ ):

Equation 16.

$$y(k)=g(k)+h(k)$$

High-order outputs (for N=8, outputs  $k=\{4,5,6,7\}$ ):

15 Equation 17.

$$\hat{y}(k)=y(N-1-k)=g(k)-h(k)$$

Note that  $g(k)$  operates directly on even input values to yield output values directly, whereas  $h(k)$  involves grouping of input values, as well as multiplication by the values  $d1$ ,  $d3$ ,  $d5$  and  $d7$ .

20 As always, the designer of an IDCT circuit is faced with a number of trade-offs, such as size versus speed and greater number of implemented devices versus reduced interconnection complexity. For example, it is often possible to improve the speed of computation by including additional, or more complicated devices on the silicon chip, but this obviously makes the implementation bigger or more complex. Also, what is available or desired on the IDCT chip may limit or preclude the use of sophisticated, complicated, designs such as "look-ahead" adders.

#### 25 Standards of Accuracy

Assuming infinite precision and accuracy of all calculations, and, thus, unlimited storage space and calculation time, the image recreated by performing the IDCT and DCT-transformed image data would reproduce the original image perfectly. Of course, such perfection is not to be had using existing technology.

In order to achieve some standardization, however, IDCT systems are at present measured according to a standardized method put forth by the Comité Consultatif International Telegraphique et Telephonique ("CCIT") in Annex 1 of

35

CCITT Recommendations H.261 - Inverse Transform Accuracy Specification.

This test specifies that sets of 10,000 8-by-8 Blocks containing random integers be generated. These blocks are then DCT and IDCT transformed (preceded or followed by predefined rounding, clipping and arithmetic operations) using predefined precision to produce 10,000 sets of 8-by-8 'reference' IDCT output data.

When testing an IDCT implementation, the CCITT test blocks are used as inputs. The actual IDCT transformed outputs are then compared statistically with the known 'reference' IDCT output data. Maximum values are specified for the IDCT in terms of peak, mean, mean square, and mean mean error of blocks as a whole and as individual elements. Furthermore, the IDCT must produce all zeros output if the corresponding input block contains all zeros, and the IDCT must meet the same standards when the sign of all input data is changed. Implementations of the IDCT are said to have acceptable accuracy only if their maximum errors do not exceed the specified maximum values when these tests are run.

Other known standards are those of the Institute of Electrical and Electronic Engineers ('IEEE'), in 'IEEE Draft Standard Specification for the Implementation of 8 by 8 Discrete Cosine Transform', P1180/D2, July 18, 1990; and Annex A of '8 by 8 Inverse Discrete Cosine Transform', ISO committee Draft CD 11172-

2. These standards are essentially identical to the CCITT standard described above.

#### Hardware Implementation

FIG 9 is a simplified block diagram illustrating the data flow of the IDCT method according to one embodiment of the present invention (although the hardware structure, as is illustrated and explained below, is made more compact and efficient). In FIG 9, the inputs to the system such as Y[0] and Y[4], and the outputs from the system, such as X[3] and X[6], are shown as being conveyed on single lines. It is to be understood that each of the single-drawn lines in FIG 9 represents several conductors in the form of data buses to convey, preferably in parallel, the several-bit wide data words to which each input and output corresponds.

In FIG 9, the large open circles 225 and 226 represent two-input adders, whereby a small circle 227 at the connection point of an input with the adder indicates that the complement of the corresponding input word is used. Adders with such a complementing input, thus, subtract the complemented input from

the non complemented input. For example, although the output T0 from the upper left adder will be equal to  $Y[0] + Y[4]$  that is,  $T0 = Y0 + Y4$ , the adder with the output T1 forms the value  $Y0 + (-1) \cdot Y4 = Y0 - Y4$ . Adders with a single complementing input can, therefore, be said to be differencing components.

5 Also in FIG 9, constant-coefficient multipliers are represented by solid triangles 230 in the data path. For example, the input Y1 passes through a square root of two multiplier before entering the adder to form B0. Consequently, the intermediate value  $T3 = Y2.T3 = Y2.c1s + Y6.c3s$ , and the intermediate value  $B2 = pl.c3s - p3.c1s = (Y1 + Y3).c3s - (Y5 + Y7).c1s$ . By performing the indicated  
10 additions, subtractions, and multiplications, one will see that the illustrated structure implements the expressions E11 and E12 for  $g(0)$  to  $g(3)$  and  $h(0)$  to  $h(3)$ .

FIG 9 illustrates an important advantage of the embodiment, in accordance with the present invention. As FIG 9 shows, the structure is divided into  
15 four main regions: a pre-common block, PREC 231, that forms the paired inputs  $\hat{p}(k)$  and multiplies the input  $y(1)$  by the square root of two; a first post-common block, POSTC1 233, that includes four multipliers for the constants  $d1$ ,  $d3$ ,  $d5$ ,  $d7$  (see expression E12); a second post-common block, POSTC2 235, that sums the  $g0$  to  $g3$  terms and the  $h0$  to  $h3$  terms for the low order outputs,  
20 and forms the difference of the  $g0$  to  $g3$  terms and the  $h0$  to  $h3$  terms for the high-order outputs (See expressions E17 and E17); and a common block, CBLK 232, is included in both the even and odd data paths. In the processing circuitry according to the embodiment of the present invention, the common operations performed on the odd and even numbered inputs are carried out by a single  
25 structure, rather than duplicated structure as illustrated in FIG 9.

To understand the method of operation and the advantages of certain digital structures used in the embodiment, it is helpful to understand what "carry bits". As a simple example, note that the addition of two binary numbers is such that  $1 + 1 = 0$ , with a carry of "1", which must be added into the next higher order  
30 bit to produce the correct result "10" (the binary representation of the decimal number "2"). In other words,  $01 + 01 = 00$  (the "sum" without carry) + 10 (the carry word); adding the "sum" to the "carry word" one gets the correct answer  $00 + 10 = 10$ .

As a decimal example, assume that one needs to add the numbers '436' and '825'. The common procedure for adding two numbers by hand typically proceeds as follows:

1. Units '6' plus '5' is '11' with a carry of '1' into the 'tens' position -  
Sum: 1, Carry-In: 0, Carry-Out: 0.
2. Tens: '3' plus '2' is '5', plus the '1' carried from the preceding step, gives '6' with no carry -  
Sum: 5, Carry-In: 0, Carry-Out: 0.
3. Hundreds: '4' plus '8' is '12' with a carry of 1 into the thousands, but with no carry to be added in from the previous step;  
Sum: 2, Carry-In: 1, Carry-Out: 1.
4. Thousands: '0' plus '0', plus the '1' carried from the hundreds gives, '1'  
Sum: 0, Carry-In: 1, Carry-Out: 0.

The answer, '1261', is, thus, formed by adding the carry-in sum for each position to the sum for the same position, with the carry-in to each position being the carry-out of the adjacent lower-order position. (Note that this implies that the carry-in to the lowest order position is always a '0'). The problem, of course, is that one must wait to add the '4' and '8' in the hundreds place until one knows whether there will be a carry-in from the tens place. This illustrates a "ripple adder", which operates essentially in this way. A ripple adder, thus, achieves a 'final' answer without needing extra storage elements, but it is slower than some other designs.

One such alternative design is known as 'carry-save', in which the sum of two numbers for each position is formed by storing a partial sum or result word (in this example, 0251) and the carry values in a different word (here, 1010). The full answer is then obtained by 'resolving' the sum and carry words in a following addition step, thus,  $0251 + 1010 = 1261$ . Note that one can perform the addition for every position at the same time, without having to wait to determine whether a carry word can be added to the partial result at any time as long as it is saved.

Since the resolving operations typically require the largest proportion of the time required in each calculation stage, speeding up these operations has a significant effect on the overall operating speed while requiring only a relatively small increase in the size of the transform. Carry-save multipliers, therefore, are usually faster than those that use ripple adders in each row. However, this gain in time comes at the cost of greater complexity, since the carry word for

each addition in the multiplier must be either stored or passed down to the next addition. Furthermore, in order to obtain the final product of a multiplication, the final partial sum and final carry word will have to be resolved, normally by addition in a ripple adder. Note, however, that only one ripple adder will be needed, so that the time savings are normally proportional to the size of the multiplication that must be performed. Furthermore, note that a carry word may be treated as any other number to be added in and as long as it is added in at some time before the final multiplication answer is needed, the actual addition can be delayed.

In this embodiment of the present invention, this possibility of delaying resolution is used to simplify the design and to increase the throughput of the IDCT circuitry. Also, certain bits of preselected carry words are, optionally and deliberately forced to predetermined values before resolution in order to provide greater expected accuracy of the IDCT result based on a statistical analysis of test runs of the invention on standard test data sets.

FIG 10 is a block diagram that illustrates a preferred structure, in accordance with the present invention. In this preferred embodiment of the present invention, the even and odd numbered inputs are time-multiplexed and are processed separately in the common block CBLK 232. The inputs may be processed in either order.

In FIG 10, the notation  $Y[1,0]$ ,  $Y[5,4]$ ,  $Y[3,2]$  and  $Y[7,6]$  is used to indicate that the odd numbered inputs Y1, Y3, Y5, Y7 preferably pass through the calculation circuitry first, followed by the even numbered inputs Y0, Y2, Y4, Y6. This order is not essential to the present embodiment; nonetheless, as is explained below, certain downstream arithmetic operations are performed only on the odd numbered inputs, and by entering the odd numbered input values first, these downstream operations can be processing at the same time that arithmetic operations common to all inputs are performed upstream on the even numbered inputs. This reduces the time that several arithmetic devices would otherwise remain idle.

Similarly, the notation  $X[0,7]$ ,  $X[1,6]$ ,  $X[3,4]$ ,  $X[2,5]$  is used to indicate that the low order outputs X0, X1, X2, X3 are output first, followed by the high order outputs X4, X5, X6, X7. As FIGS. 9 and 10 illustrate, the inputs are preferably initially not grouped in ascending order, although this is not necessary since to odd numbered inputs are Y1, Y5, Y3, and Y7. Arranging the input signals in this

order makes possible the simple "butterfly" data path structure shown in FIGS. 9 and 10 and greatly increases the interconnection efficiency of the implementation of the present invention in silicon semiconductor devices.

As shown in FIG 10, adders and subtractors are indicated by circles either a '+' (adder) 235, '-' (subtractor) 236 which is an adder with one complementing input or '±' (resolving adder/subtractor, which is able to switch between addition and subtraction 237). The left most adders and subtractors in the common block 232 of the two m-bit input words is the m-bit partial resulting parallel with the m-bit or (m-1) bit word containing the carry bits of the addition/subtraction. In other words, the first additions and subtractions in the common block CBLK 232 are preferably unresolved, meaning that the addition of the carry bits is delayed until a subsequent processing stage. The advantage of this step is that such carry-save adder/subtractors since they do not need to perform the final addition of the carry-bit word to the result. Resolving adders may, however, also be used in order to reduce the bus width at the outputs of the adders.

FIG 10 also illustrates the use of one and two input latches in the preferred embodiment of the present invention. In FIG 10, latches are illustrated as rectangles 238 and are used in both the pre-common block PREC 231 and the post-common block POSTC 233. Single-input latches are used at the inputs of the multipliers D1, D3, D5 and D7, as well as to latch the inputs to the resolving adders/subtractors which are the computed  $g(k)$  and  $h(k)$  values corresponding to the respective outputs from latches  $g[0,7]$ ,  $g[1,6]$ ,  $g[3,4]$  and  $g[2,5]$  and  $h[0,7]$ ,  $h[1,6]$ ,  $h[3,4]$  and  $h[2,5]$ . As such, the resolving adders/subtractors perform the addition or subtraction indicated in expressions E16 and E17 above.

As described previously, the even-numbered inputs Y0, Y2, Y4 and Y6 do not need to be paired before being processed in the common block CBLK 232. However, not only do the odd-numbered inputs require such pairing, but the input Y12 must also be multiplied by the square root of two in order to ensure that proper input values are presented to the common block CBLK 232. The pre-common block PREC 231, therefore, includes a 2-input multiplexing ("mux") latch C10, C54, C32 and C76 for each input value. One input to the 2-input mux latch is consequently tied directly to the unprocessed input values, whereas the other input is received from the resolving adders and, for the input Y1, the resolving square root of two multiplier. The correct paired or unpaired inputs can, therefore, be easily presented to the common block CBLK 232 easily by simple

switching of the multiplexing latches between their two inputs.

As FIG 10 illustrates, the square root of two multipliers D1, D3, D5, D7 preferably resolve their outputs, that is, they generate results in which the carry bits have been added in to generate a complete sum. This ensures that the outputs from the multipliers have the same bus width as the un-multiplied inputs in the corresponding parallel data paths.

The preferred embodiment of the common block 232, in accordance with the present invention, also includes one 'dummy' subtractor 240 in the forward data path for Y[1,0] and Y[5,4], respectively. These devices act to combine the two inputs (in the case of the dummy subtractor, after 2's-complementing the one input) in such a way that they are passed as parallel outputs. In each case, the one input is manipulated as if it contained carry bits, which are added on in the subsequent processing stage. The corresponding addition and subtraction is, thus, performed, although it is delayed.

This technique reduces the resources required in the upper two data paths since a full-scale adder/subtractor need not be implemented for these devices. Therefore, the 'combiners' act as adders and subtractors and can be implemented for these devices and can be implemented either as simple conductors to the next device (for addition), or as a row of inverters (for subtraction), either of which requires little or no additional circuitry.

The use of such combiners also means that the outputs from the initial adders and subtractors in the common block CBLK 232 will all have the same width and will be compatible with the outputs of the carry-save adder/subtractor found in the bottom two data paths, with which they form inputs to the subsequent resolving adders and subtractors in the common block CBLK.

As described previously, the even-numbered inputs are processed separately from the odd-numbered inputs in this preferred embodiment of the present invention. Assume, further, that the odd-numbered inputs are to be processed first. Supervisory control circuitry (not shown in FIG 10) applies the odd-numbered input words to the pre-common block PREC, and selects the lower inputs (viewed as in FIG 10) of the multiplexing latches C10, C54, C32, C76 which then stores the paired values p0 to p3 (see FIG 9 and the definition of p(n) above). The latches 1h0, 1h1, 1h3 and 1h2 are then activated to latch the values H0, H1, H3 and H2, respectively.



The supervisory control circuitry latches and then selects the upper inputs of the two-input multiplexing latches C10, C54, C32 and C76 in the precommon block PREC 231 and applies the even numbered input words to these latches. Since the even-numbered inputs are used to form the values of  $g_0$  to  $g_3$ , the supervisory control circuitry also opens the latches  $Lg_0$  to  $Lg_3$  in the post-common block POSTC 233, to store the  $g(k)$  values.

Once the  $g(k)$  and  $h(k)$  values are latched, the post-common block POSTC 233 outputs the high-order signals  $X_7$ ,  $X_6$ ,  $X_5$  and  $X_4$  by switching the resolving adder subtractors to the subtraction mode. The low order output signals  $X_3$ ,  $X_2$ ,  $X_1$  and  $X_0$  are then generated by switching the resolving adders/subtractors to the addition mode. Note that the output data can be presented in an arbitrary order, including natural order.

The preferred multiplexed implementation, in accordance with the present invention, is illustrated in greatly simplified, schematic form in FIG 10, performs the same calculations as the non-multiplexed structure illustrated in FIG 9. The number of adders, subtractors and multipliers in the common block CBLK 232 is, however, cut in half and the use of dummy adder/subtractors 240 further reduces the complexity of the costly arithmetic circuitry.

FIG 11 illustrates the main components and data lines of an actual implementation of the IDCT circuit according to the embodiment of the present invention. The main components include the precommon block circuit PREC 231, the common block circuit CBLK 232, and the post-common block POSTC 233. The system also includes a controller CNTL 241 that either directly or indirectly applies input, timing and control signals to the precommon block PREC 231 and post-common block POSTC 233.

In the preferred embodiment of the present invention, the input and output signals ( $Y_0$  to  $Y_7$  and  $X_0$  to  $X_7$ , respectively) are 22 bits wide. Tests have indicated that this is the minimum width that is possible which still yields acceptable accuracy as measured by existing industry standards. As is explained in greater detail below, this minimum width is achieved in part by deliberately forcing certain carry words in selected arithmetic devices to be either a '1' or a '0'. This bit manipulation, corresponding to an adjustment of certain data words, is carried out as the result of a statistical analysis of the results of the IDCT system, in accordance with the present invention, to the after using the IDCT transformation of known input test data. By forcing certain bits to predetermined values,

it was discovered that the effects of rounding and truncation errors could be reduced, so that the spatial output data from the IDCT system could be made to deviate less from the known 'correct' spatial data. The present invention is equally applicable, however, to other data word lengths since the components used in the circuit according to the present embodiment can all be adapted to different bus widths using known methods.

Although all four inputs that are processed together could be input simultaneously to the pre-common block PREC along 88 parallel conductors (4 x 22), pixel words are typically converted one at a time from the transmission data. According to the present embodiment, input data words are, therefore, preferably all conveyed serially over a single 22 bit input bus and each input word is sequentially latched at the proper input point in the data path. As shown in FIG 11, the 22 bit input data bus is labelled T\_IN[21:0] 242.

In the Figures and in the discussion below, the widths of multiple-bit signals are indicated in brackets with the high-order bit to the left of a colon ':' and the least significant bit (LSB) to the right of the colon. For example, the input signal T\_IN[21:0] 242 is 22 bits wide, with the bits being numbered from 0 to 21. A single bit is identified as a single number within square brackets, thus, T\_IN[1] indicates the next to least significant bit of the signal T\_IN.

The following control signals are used to control the operation of the pre-common block PREC 231 in the preferred embodiment of the present invention.

IN\_CLK, OUT\_CLK: The system, in accordance with the present invention, preferably uses a non-overlapping two phase clock. The signals IN\_CLK and OUT\_CLK are accordingly columns of latches that hold the values of input, intermediate, and output signals.

LATCH10, LATCH54, LATCH32, LATCH76: Preferably, one 22-bit word is input to the system at a time. On the other hand, four input signals are processed at a time. Each input signal must, therefore, be latched at its appropriate place in the architecture before being processed with three other input words. These latch signals are used to enable the respective input latches. The signal LATCH54, for example, is first used to latch input signal Y5 and later to latch input signal Y4, which enters the pre-common block PREC 231 at the same point as the input signal Y5 (see FIG 10) but during a subsequent processing stage.

LATCH: Once the four even or odd-numbered input signals are latched into the pre-common block PREC 231, they are preferably shifted at the same

time to a subsequent column of latches. The signal LATCH is used to enable a second column of input latches that hold the four input values to be operated on by the arithmetic devices in the pre-common block PREC 231.

SEL\_BYP, SEL\_P: As FIG 10 illustrates, the even-numbered input signals that are latched into the latches C10, C54, C32 and C76 should be those that bypass the adders and the square root of two resolving multiplier. The odd-numbered input signals, however, must first be paired to form the paired inputs  $p(n)$ , and the signal Y1 must be multiplied by the square root of two. The control signal SEL\_P is activated in order to select the paired input signals. Hence, these signals are used to control gates that act as multiplexers to let the correct signals pass to the output latches of the precommon block PREC 231.

As discussed previously, not having to arrange the inputs in strictly ascending order leads to a simplified "butterfly" bus structure with high interconnection efficiency. As also described, the odd inputs are preferably applied as a group to the pre-common block first, followed by the even-numbered inputs, but any order may be used within each odd or even group, i.e., any order of inputs may be used, however, suitable latch arrangements as separately provided to process the odd-numbered inputs, or at least are provided in separate regions of the circuit.

The supervisory control circuitry also generates timing and control signals for the post-common block POSTC 233. These control signals are as follows:

EN\_BH, EN\_GH: Referring again to FIG 9, the outputs from the common block CBLK 232, after processing of the odd-numbered inputs, are shown as H0, H1, H3, and H2. These signals are then sent to the coefficient multipliers, d1, d3, d7, d5, respectively, in the first post common block POSTC 1 233. The signal EN\_BH is used to enable latches that hold the g0 to g3 values, as well as to enable the latches that hold the h0 to h3 values after they have been multiplied in the coefficient multipliers.

ADD, SUB: As FIG 10 illustrates, the embodiment includes a bank of resolving adders/subtractors that sum and difference  $k$  and  $h(k)$  values in order to form the low-order outputs, respectively. The signals ADD, SUB are used to set the resolving adders/subtractors in the addition and subtraction modes, respectively.

EN\_O: This signal is used to enable output latches that latch the results from the resolving adders/subtractors.

MUX\_OUT0, MUX\_OUT61, MUX\_OUT43, MUX\_OUT52: In accordance with the present invention, the output data from the system is preferably transmitted over a single 22-bit output bus, so that only one output value (X0 to X7) is transferred at a time. These signals are activated sequentially to select which of the four latched output values is to be latched into a final output latch. Accordingly, these signals thus act as the control signals for a 4-to-1 multiplexer.

T\_OUT[21:0]: This label indicates the 22-bit output signal from the post-common block POSTC 233.

The output signals from the pre-common block PREC 231 are latched to form the input signals to the common block CBLK 232. As shown in FIG 11, the output signals from the pre-common block PREC 231 are presented as the four 22-bit data words C10[21:0], C154[21:0], C132[21:0], C178[21:0], which become the input signals IN[0], IN[1], IN[3], IN[2], respectively, to the common block CBLK 232.

As FIG 11 shows, the four 22-bit results from the common block CBLK 232 are transferred in parallel as output signals OUT0[21:0], OUT1[21:0], OUT3[21:0], OUT2[21:0], which are then latched as the input signals of the post-common block POSTC 233 as CO70[20:1], CO61[21:0], CO43[21:0], CO52[21:0].

One should take particular note that no control signals are required for the common block CBLK. Because of the unique structure of the IDCT system in this example, the common block of the system's operations can be performed as pure logic operations, with no need for clock, timing or control signals. This further reduces the complexity of the device. One should also note that in certain applications (particularly those in which there is plenty of time to perform all needed arithmetic operations) the pre-common and post-common blocks PREC 231, POSTC 233 may also be arranged to operate without clock timing or control signals.

FIG 12 is a block diagram of the pre-common block PREC 231 of the present invention. In this and following Figures, the notation 'S1[a], S2[b], ..., SM[Z]', where S is an arbitrary signal label and a, b, ..., z are integers within the range of the signal's bus width, indicates that the selected bits a, b, ..., z from the signals S1, S2, ..., SM are transferred in parallel over the same bus, with the most significant bits (MSBs) being the selected bits 'a' of the signal S1, and the least significant bits (LSBs) being the selected 'z' of signal SM. The

selected bits do not have to be individual bits, but rather, entire or partial multi-bit words may also be transmitted along with other single bits or complete or partial multi-bit words. In the Figures, the symbol S will be replaced by the corresponding signal label.

5 For example, in FIG 12, a square root of two multiplier is shown as R2MUL. The 'save' or 'unresolved sum' output from this non-resolving multiplier is indicated as the 21-bit word M5S[20:0], similarly, the 'carry' output from the multiplier R2MUL is shown as the 22-bit word M5C[20:0], which is transferred  
10 over the bus to the 'b' input of a carry-save resolving adder M5A. (Recall that a '0' is inserted as an MSB to the least significant 21 bits of the save output, however, this is accomplished before being applied to the 'a' input of the resolving adder M5A. This is indicated in FIG 12 by the notation GND.M5S[20:0]). In other words the conductor corresponding to the MSB Input to the adder M5A is forced to be a '0' by tying it to ground GND.

15 In order to understand why a '0' is inserted as the 22nd bit of the 'sum', observe that if the partial sum of a multiplication is n places wide, the carry word is shifted one place to the left relative to the partial sum. The carry word, therefore, extends to n + 1 places with a valid data bit in the n + 1th position with an '0' in the least significant position (since there is nothing before this  
20 position to produce a carry bit into the units position). If these two words are used as inputs to a resolving binary adder, care must be taken to ensure that the bits (digits) of the carry word are properly aligned with the corresponding bits of the partial sum. This also ensures that the decimal point (even if only implied, as in integer arithmetic) is kept 'aligned' in both words. Assuming the inputs to the  
25 adder are n + 1 bits wide, a '0' can then be inserted into the highest-order bit of all n-bit positive partial sum words to provide an n + 1 bit input that is aligned with the carry word at the other input.

As is described above previously, the four inputs that are processed at a given time in the pre-common block PREC 231 are transferred over the input bus  
30 T\_IN(21:0). This input bus is connected to the inputs of four input latches IN10L, IN54L, IN32L, AND IN76L. Each respective latch is enabled only when the input clock signal IN\_CLK and the corresponding latch selection signal LATCH10, LATCH54, LATCH32, LATCH76 are high. The four inputs can, therefore, be  
35 latched into their respective input latches in four periods of the IN\_CLK signal by sequential activation of the latch enabling signals LATCH10, LATCH54,

LATCH32, and LATCH76. During this time, the LATCH signal should be low (or on a different phase) to enable the input latches IN10L, IN54L, IN32L, and IN76L to stabilize and latch the four input values.

An example of the timing of the latches, in accordance with the present invention, is illustrated in FIG 13. Once the four input signals are latched in the preferred order, they are passed to a second bank of latches L10L, L54L, L32L, L76L. These second bank of latches are enabled when the signals OUT\_CLK and LATCH are high. This signal timing is also illustrated in FIG 13.

Note that the system of the present invention does not have to delay receipt of all eight input words. Once all the even or odd input words are received and latched in IN10L, IN54L, IN32L and L76L, this frees the In latches, which can then begin to receive the other four input signals without delay at the next rising edge of IN\_CLK.

The 2-digit suffix notation [10, 54, 32, 76] used for the various components illustrated in the Figures indicates that odd-numbered signals are processed first, followed by the even-numbered signals on a subsequent pass through the structure. As is mentioned above, this order is not required by the present invention, and it will be appreciated by one of ordinary skill in the art that additional orders may be used.

Once the four input signals are latched in proper order in the second set of latches L10L, L54L, L32L, L76L, the corresponding values are either passed as inputs to output latches C10L, C54L, C32L and C76L on activation of the selected bypass signal SEL\_BYP, or they are passed as paired and multiplexed inputs to the same output latches upon activation of the 'select p' signal SEL\_P. In other words, all signals are passed, both directly and indirectly, via arithmetic devices, to the output latches C10L, C54L, C32L, C76L of the pre-common block PREC 231. The proper values, however, are loaded into these latches by activation of the 'select bypass' signal SEL\_BYP (for even-numbered inputs Y0, Y2, Y4, and Y6) or the 'select p' signal SEL\_P (for the odd-numbered inputs Y1, Y3, Y5 and Y7). As will be appreciated by one of ordinary skill in the art, the desired timing and order of these and other control signals is easily accomplished in a known manner by proper configuration and/or [micro-] programming of the controller CNTL 241.

The uppermost input value at the output of latch L10L is passed first to the square root of two-multiplier R2MUL and then to the resolving adder M5A as

indicated. The output from the resolving adder M5A is shown as an equivalent of the resolved multiplication of the output from the latch L10L by the square root of two. The outputs from the other three latches L54L, L32L, L76L are also transferred to corresponding output latches C54L, C32L and C76L, respectively, both directly via 22-bit latch buses LCH54[21:0], LCH32[21:0] LCH76[21:0] and indirectly to the output latches via resolving adders P2A, P1A and P3A, respectively.

In the present invention, each resolving adder P2A, P1A, P3A has two inputs "a" and "b". For adder, P2A, the one input is received from the latch L32L, and the other input is received from the latch L54L. For input values Y5 (latched in L54L) and Y3 (latched in L32L), the output from the adder P2A will, therefore, be equal to  $Y5 + Y3$ , which, as is shown above, is equal to  $p(2)$ . Hence, the adders "pair" the odd-numbered inputs to form the paired input values  $p(1)$ ,  $p(2)$  and  $p(3)$ . Of course, the even-numbered input signals latched in L54L, L32L, and L76L will also pass through the resolving adders P2A, P1A and P3A, respectively, however, the resulting  $p$  "values" will not be passed to the output latches C54L, C32L and C76L because the "select  $p$ " signal SEL\_P will not be activated for even-numbered inputs.

The values that are latched in the output latches C10L, C54L, C32L and C76L upon activation of the Input clock signal IN\_CLK will therefore be equal to either the even-numbered inputs Y0, Y2, Y4, Y6 or the paired input values P0, P1, P2, P3 for the odd-numbered inputs. One should recall that the input Y(1) is "paired" with the value U(-1), which is assumed to be zero. As illustrated in FIG 12, this assumption is implemented by not adding anything to the value Y1. Instead, Y1 is only multiplied by the square root of two as is shown in FIGS. 9 and 10.

FIG 14 illustrates the preferred architecture of the common block CBLK 232, in accordance with the present invention. Because of the various multiplications and additions in the different system blocks, it is necessary or advantageous to scale down the input values to the common block before performing the various calculations. This ensures a uniform position for the decimal point (which is implied for integer arithmetic) for corresponding inputs to the various arithmetic devices in the system.

Accordingly, the input values IN0[21:0] AND IN1[21:0] are accordingly scaled down by a factor of four, which corresponds in digital arithmetic to a right

shift of two bits. In order to preserve the sign of the number (keep positive values positive and negative values negative) in binary representation, the most significant bit (MSB) must then be replicated in the two most significant bits of the resulting right-shifted word; this process is known as "sign extension". Hence, the input value IN0 is downshifted by two bits with sign extension to form the shifted input value indicated as IN[21], INQ[21], INQ[21:2]. The input value IN\*[21:0] is similarly sign-extended two places. The input IN2 is also shifted and extended to form IN2[21], IN2[21:1]. These one-position shifts correspond to truncated division by a factor of two.

As shown in FIG 10, the input IN2, IN3 are those which must be multiplied by the scaled coefficients c1s and c3s. Each input IN3 and IN2 must be multiplied by each of the scaled coefficients. As FIG 14 illustrates, this is implemented by the four constant-coefficient carry-save multipliers MULC 1S, MULNC1S, MULC3S3, and MULC2S2. One should note that the bottom multiplier for IN2 is an inverting multiplier MULC1S, that is, its output corresponds to the negative of the value of the input multiplied by the constant C1S. Therefore, the value latched in C75 is subtracted from the value latched in C32 (after multiplication by C3S). By providing the inverting multiplier MULNC1S, subtraction is implemented by adding the negative of the corresponding value, which is equivalent to forming a difference. This allows the use of identical circuitry for the subsequent adders, while allowing a non-inverting multiplier may be used with a following subtractor.

In the illustrated embodiment of the present invention, four cosine coefficient multipliers MULC1S, MULNC1S, MULC2S3, and MULC3S2 are included. If arrangements are made for signals to pass separately through the multipliers, however, the necessary multiplications can be implemented using only two multipliers, one for the c1s coefficient and one for the c3s coefficient.

In accordance with the present invention, the multipliers for MULC1S, MULNC1S, MUL3S3 and MULC3S2 are preferably of the carry-save type, which means that they produce two output words, one corresponding to the result of the various rows of additions performed within a hardware multiplier, and another corresponding to the carry bits generated. The outputs from the multipliers are then connected as inputs to either of two 4-input resolving adders BT2, BT3.

For ease of illustration only, five of the output buses from the multipliers are not drawn connected to the corresponding input buses of the adders, as will



be appreciated by one of ordinary skill in the art, these connections are to be understood, and are illustrated by each respective output and input having the same label. Hence, the save output M1S[20:0] of the multiplier MULC1S is connected to the lower 21 bits of the "save-a" of the adder BT3.

As shown in FIG 14, five of the inputs to the adders BT2 and BT3 are shown as being "split". For example, the "ca" input of the adder BT2 is shown as having IN3[2:1] over M3C[20:0] being input as the least significant 21 bits. Similarly, the "sa" (the "save-a" input) of the same adder is shown as being GND, GND over M3S[19:0]. This means that two zeros are appended as the two most significant bits of this input word. Such appended bits ensure that the proper 22-bit wide input words are formed with the proper sign.

The carry-save adders BT2 and BT3 add the carry and save words of two different 22-bit inputs to form a 22-bit output save word T3S[21:0] and a 21-bit output carry word T3C[21:1]. Accordingly, the input to each adder is thus 55 bits wide and the output from each adder is 43 bits wide. As FIG 10 illustrates, the output from the latch C10 is combined with the output from the latch C54 in the upper-most data path before addition with the output from the carry-save adder BT3. The "combination" is not, however, necessary until reaching the following adder in the upper data path. Consequently, as FIG 14 shows, the shifted and sign-extended input value IN0 is connected to the upper carry input.

The upper carry input of adder CS0 is connected to the shifted and sign-extended input value IN0, and the shifted and sign-extended input IN1 is connected as the upper save input of the same adder. In other words, IN0 and IN1 are added later in the adder CS0.

The designation "dummy" adder/subtractor 240 used in FIG. 10, therefore, indicates which operation must be performed, although it does not necessarily have to be performed at the point indicated in FIG. 10. Similarly, the lower dummy subtractor 240 shown in FIG. 10 requires that the output from latch C54 be subtracted from the output from latch C10. This is the same as adding the output from C10 to the complement of the output of C54.

Referring once again to FIG. 14, the complement of the input IN1 (corresponding to the output of latch C54 in FIG 10) is performed by a 22-bit input inverter IN1[21:0] (which generates the logical inverse of each bit of its input, bit-for-bit). The complement of IN1 value—NIN1[21:0]—is passed to the upper "save" input of the adder CS1, with the corresponding upper "carry" input being the

shifted and sign-extended IN0. The upper portion of the adder CS1, therefore, performs the subtraction corresponding to IN0 minus IN1.

In the lower two data paths shown in FIG. 10, resolving subtractors are used instead of the resolving adders shown in the upper two data paths at the output of the common block CBLK 232. Each resolving adder or subtractor is equivalent to a carry-save adder or subtractor followed by a resolving adder. This is shown in FIG. 14. Subtractors CS2 and CS3 have as their inputs the processed values of IN0 to IN3 according to the connection structure shown in FIG. 10.

The 22-bit carry and save outputs from each of the adders/subtractors C20-CS3 are resolved in the resolving adders RES0-RES3. As will be appreciated by one of ordinary skill in the art, resolution of carry and save outputs is well understood in the art of digital design and is, therefore, not described in greater detail here. As FIG. 14 illustrates, the save outputs the carry-save adders/subtractors CS0-CS3 are passed directly as 22-bit inputs to the "a"-input of the corresponding resolving adders RES0-RES3.

As is also well known in the art, the 2's-complement of a binary number is formed by inverting each of its bits (changing all "1"s to "0"s and vice versa) and then adding "1". Note that the "1" can be added immediately after the bit inversion, or later. The LSB of a carry word will always be a "0" which is implemented in the illustrated embodiment of the present invention by tying the LSB of the carry words O0C and O1C to ground GND as they are input to the resolving adders RES0 and RES1, respectively. The addition of "1" to the carry outputs of the subtractors CS2 and CS3 to form 2's-complemented values, however, is implemented by tying the LSB of these data words O2C and O3C to supply voltage VDD, thus "replacing" the "0" LSB of the carry word by a "1", which is equivalent to addition by "1".

For the reasons provided above, a "0" is appended as the LSB to the 21-bit carry words from the carry-save adders CS0 and CS1 (by tying the LSB to ground GND) and the LSB of the carry words from the carry-save subtractors CS2 and CS3 is set equal to "one" by tying the corresponding data line to the supply voltage VDD. The resolving adders RES0-RES3, therefore, resolve the outputs from the adders/subtractors CS0-CS3 to form the 22-bit output signals OUT0[21:0] - OUT3[21:0].

Two advantages of the IDCT circuitry according to the embodiment of the present invention can be seen in FIG 14. First, no control or timing signals are required for the common block CBLK 232. Rather, the input signals to the common block are already processed in such a way they can be applied immediately to the pure-logic arithmetic device in the common block 232. Second, by proper scaling of the data words, integer arithmetic can be used throughout (or, at least, decimal point for all values will be fixed). This avoids the complexity and slowness of floating-point devices, with no unacceptable sacrifice of precision.

Yet another advantage of the embodiment of the present invention is that, by ordering the inputs as shown, and by using the balanced decimated method in accordance with the present invention, similar design structures can be used at several points in the silicon implementation. For example, as shown in FIG. 14, the constant coefficient multipliers MULC1S, MULC3S3, MULC3S2 and MULNC1S all have similar structures and receive data at the same point in the data path, so that all four multipliers can be working at the same time. This eliminates "bottlenecks" and the semiconductor implementation is, therefore, able to take full advantage of the duplicative, parallel structure. The carry-save adders BT2 and BT3 similarly will be able to work simultaneously, as will the following carry-save adders and subtractors. This symmetry of design and efficient simultaneous utilization of several devices is common throughout the structure according to the embodiment of the present invention.

FIG 15 shows the preferred arrangement of the post-common block POSTC 233 in accordance with the present invention. As FIG. 10 shows, the primary functions of the post-common POSTC 233 are to form the  $h_0$  to  $h_3$  values by multiplying the outputs of the common block by the coefficients  $d_1$ ,  $d_3$ ,  $d_5$  and  $d_7$ ; to add the  $g(k)$  and  $h(k)$  values to form the low order outputs; and to subtract the  $h(k)$  values from the corresponding  $g(k)$  values to form the high-order outputs. Referring now to both FIG. 10 and FIG. 15, the post-common block POSTC 233 latches the corresponding outputs from the common block CBLK 232 into latches BH0L, BH1L, BH3L and BH2L when the Bh latches are enabled, the control circuitry sets the EN\_BH signal high, and the output clock signal OUTC\_CLK signal goes high. The  $g(k)$ ,  $g_0$  to  $g_3$  values are latched into corresponding latches G0L, G1L, G3L and G2L when the control circuitry enables these latches via the signal EN\_GH and input clock signal IN\_CLK goes high.

The processed odd-numbered inputs, that is, the values  $H0$  to  $H3$ , are latched into latches  $H0L$ ,  $H1L$ ,  $H3L$  and  $H2L$  when the  $EN\_GH$  and  $IN\_CLK$  signals are high, via the constant coefficient multipliers  $D1MUL$ ,  $D3MUL$ ,  $D5MUL$  and  $D7MUL$ . These multipliers multiply, respectively by  $d1$ ,  $d3$ ,  $d5$  and  $d7$ . In the preferred embodiment, these constant-coefficient multipliers are preferably carry-save multipliers in order to simplify the design and to increase calculation speed. As FIG 15 illustrates, the "carry" ("c") outputs from the constant coefficient multipliers are connected, with certain changes described below, to the  $a$  inputs of resolving adders  $H0A$ ,  $H1A$ ,  $H3A$  and  $H2A$ . The "save" ("s") outputs from the coefficient multipliers are similarly, with certain forced changes described below, connected to other input of the corresponding resolving adder.

As FIG 15 further illustrates, the LSB of the  $H0$  signal is preferably forced to be a "1" by tying the corresponding "save" output for  $H0$  is set to 0 (tied to ground GND), and the second bit (corresponding to  $H0S[1]$ ) is set to "1". The data words from the carry and save outputs of the constant-coefficient multiplier  $D3MUL$  are similarly manipulated an input to the resolving adder  $H1A$ . The advantage of these manipulations and their input to the resolving adder  $H1A$ .

In accordance with the present invention, all 22-bits of the carry output from the coefficient multipliers  $D7MUL$  and  $D5MUL$  are connected directly to the "a" input of corresponding resolving adders  $H3A$  and  $H2A$ . The MSB of each multiplier's "save" output, however, is forced to "0" by tying the corresponding data line to ground GND.

The IDCT system described was tested against the CCITT specification described above. Because of the scaling and other well-known properties of digital adders and multipliers, some precision is typically lost in the 10,000 sample, but run that forcing the various bits described above to either "0" or "1" reduced the expected error of the digital transformation. As a result of the bit manipulation of the data words, the embodiment of the present invention achieved acceptable accuracy under the CCITT standard using only 22-bit wide data words, whereas 24 bits would normally be required to produce equivalent accuracy.

Because of limited precision, and truncation and rounding errors, there is typically some inaccuracy in every data word in an IDCT system. However, forcing selected bits of a data word it was discovered that the error thereby systematically introduced into a particular data word at a particular point in the

hardware yielded statistically better overall results. Bit-forcing may also be applied "within" a multiplication, for example, by selectively forcing one or more carry bits to predetermined values.

In the present invention, the bit-forcing scheme need not be static, with certain bits always forced to take specified values, but rather a dynamic scheme may also be used. For example, selected bits of a data word may be forced to "1" or "0" depending on whether the word (or even some other data) is even or odd, positive or negative, or above or below a predetermined threshold, and the like.

Normally, only small systematic changes will be needed to improve overall statistical performance. Consequently, according to this embodiment of the present invention, the LSB's of selected data words (preferably one bit and one data word at a time, although this is not necessary) are forced to be a "1" or a "0". The CCITT test is run, and the CCITT statistics for the run are compiled. The bit is then forced to the other of "1" or "0", and the test is rerun. Then the LSB (or LSBs) of other data words are forced to "1" or "0", and similar statistics are compiled. By examining the statistics for various combinations of forced bits in various forced words, a best statistical performance can be determined.

If this statistically based improvement is not required, however, the outputs from the constant-coefficient multipliers D1MUL, D3MUL, D5MUL and D7MUL may be resolved in the conventional manner in the resolving adders H0A-H3A. The lower 21-bits of the input of the corresponding latches H0L-H3L, with the LSB of these inputs tied to ground.

The outputs from the H-latches (H0L-H3L) and the G-latches (G0L-G3L) pairwise form the respective a and b inputs to resolving adder-subtractors S70A, S61A, S43A and S52A. As was indicated above, these devise add their inputs when the ADD signal is high, and subtract the "b" input from the "a" input when the subtraction enable signal SUB is high. The second bits of the upper two latch pairs H0L, G0L, H1L and G1L are manipulated by multiplexing arrangements in a manner described below.

The outputs from the resolving adder-subtractors S70A, S61A, S43A and S52A are latched into result latched R70L, R61L, R43L, R52L.

As depicted in FIG 15b, the input words to the adder/subtractor S70A and dS61A, in accordance with the present invention, have the second bits of each input word manipulated. For example, the second bit of the input word to the "a"-

input of the adder subtractor S70A is G0[1M], G0[1M], G0[0]. In other words, the second bit is set to have the value G01M. The second bits of the other inputs to the adder/subtractors S70A and S61A are similarly manipulated. This bit manipulation is accomplished by four 2:1-bit multiplexers H01MUX, G01MUX, H11MUX and G11MUX (shown to the right in FIG 15b). In the present invention, these multiplexers are controlled by the ADD and SUB signals such that the second bit (H01M, G01M, H11M, and G11M) is set to one if the respective adder subtractor S70A, S61A is set to (ADD is high), and the second bit is set to its actual latch output value if the SUB signal is set too high. Setting of individual bits in this manner is an easily implemented high-speed operation. The preferred embodiment, therefore, includes this bit-forcing arrangement since, as is described above, statistical analysis of a large number of tests pixel words has indicated that more accurate results are thereby obtained. It is not necessary, however, to manipulate the second bits in this manner, although it gives the advantage of smaller word width.

The four high or low-order results are latched in the output latches R70L, R61L, R43L and R52L. The results are sequentially latched into the final output latched OUTF under the control of the multiplexing signals MUX\_OUT70, MUX\_OUT61, MUX\_OUT43, MUX\_OUT52. Hence, the order in which resulting signals are output can therefore be controlled simply by changing the sequence with which they are latched into the latch.

The relationship between the clock and control signals in the post-common block POSTC 233 is shown in FIGS. 13b and 13c.

As was discussed previously, two 1-dimensional IDCT operations may be performed in series, with an intervening transposition of data, in order to perform a 2-D IDCT. The output signals from the post-common block POSTC 233, are therefore, according to this embodiment of the present invention, first sorted in a known manner column-wise (or row-wise) in a conventional storage unit, such as a RAM memory circuit (not shown), and are then read from the storage unit row-wise (column-wise) so as to be passed as inputs to a subsequent pre-common block and for processing as described above in this block, and in a common block CBLK 232, and a post-common block POSTC 233.

Storing by row (column) and reading out by column (row) performs the required operation of transposing the data before the second 1-D IDCT. The output from the second POSTC 233 will be the desired, 2-D IDCT results and can

be scaled in a conventional manner by shifting to offset the scaling shifts carried out in the various processing blocks. In particular, a right shift by one position will perform the division by 2 necessary to offset the two square root of two multiplications performed in the 1-D IDCT operations.

5 Depending on the applications, this second IDCT structure (which is preferably identical to that shown FIG 11) is preferably a separate semiconductor implementation. This avoids the decrease in speed that would arise if the same circuits were used for both transforms, although separate 1-D transform imple-  
10 mentations are not necessary if the pixel-clock rate is now sufficient such that a single implementation of the circuit will be able to handle two passes in real time.

As shown in FIGS 16 through 38, a second preferred embodiment, in accordance with the present invention, uses a single one-dimensional transform. This embodiment does not require a lowering of the pixel-clock rate as discussed previously.

15 The existing "resolving-adders" in the first preferred embodiment have been changed to "fast-resolving-adders". As seen in FIG 38, these have been titled, "Fast Resolving Adders". This change has the effect of allowing more time for each datapath arithmetic block to act on its data inputs. The existing  
20 "latches" in the first preferred embodiment have been changed to 2-phase "flip-flops" or "registers".

The latching memory elements located on the front and end of the existing 1D IDCT datapath pipelines have been combined into single blocks, as shown particularly in particular in FIG 18. Additionally, the amount of memory elements present at the input and the output of the second preferred embodiment has  
25 been increased to allow variable amounts of T2 data to be buffered.

As shown in FIGS. 16 and 17, the two data streams, stream "T1" (raw unoperated upon data) and stream "T2" (data which has been through the 1D IDCT once and has been transposed in the TRAM), are introduced into the datapath pipeline in a time multiplexed fashion.

30 In the present invention, each stream takes its turn to introduce a group of data items into the datapath pipeline. The data streams are "interleaved" as they pass sequentially down the datapath pipeline and are "de-interleaved" at the datapath output, as shown in FIGS. 17, 18 and 33. A group can vary in number, but in this example, they are eight bits.

In accordance with the present invention, T1 must not be stalled. If T2 arrives at the point of interleaving with T1, but the input buffer should not introduce its data into the pipeline because this would clash with the T1 stream, then stream T2 provides an extra buffering so that T2 does not stall the data stream, but instead will buffer up data from its input stream until such a time as it may safely interleave with stream T1. This is shown in FIGs. 19 and 33 where the data from stream T1 is being loaded into the first transform in latches 0-7, using signals, "Latch 1(0) 'through' Latch 1(7)". Additionally, data from T2 is being loaded in "Latch 2(0) 'through' Latch 2(15)", as shown in FIG. 19, using signals shown in FIG. 33.

The interleaving is controlled by "T1 OK2 insert" and "T2 OK2 insert" signals. Under normal operation, the interleaving will occur when the signals go high. However, if the appropriate amount of data in the latch for T2 has not yet been reached when "T2 OK insert" goes high, then the latch will miss its opportunity and must continue buffering data until the next opportunity to insert data occurs."

In summary, if the above described buffering, in accordance with the present invention, is to occur, comparable "slippage" has to occur at the output of T2. T2 slips when it misses its data insertion point and has to continue buffering in the latches shown in FIG. 19. If T2 slipped and did not introduce data into the pipeline there will be a corresponding gap in the T2 stream output at the datapath output. This gap may be removed or "swallowed up" by use of the extra buffering at the T2 output. This process may be thought of as having a "fixed" T1 - 1D IDCT transform with a variable T2 - 1D IDCT, where the data streams are interleaved in a time multiplex fashion such that they may use the same piece of arithmetic datapath pipeline.

In the present invention, "Recovery" takes place when non-data enters T1. It is an opportunity for the T2 buffer to catch up to T1 and the datastream. Non-data is a data type that bypasses the IDCT and is shown as a data spike in "Latch 2 [  $\phi$  ]" of FIG. 34. This eventually makes its way to T2 input, which allows the T2 buffering to fill up at the output. Recovery is shown in FIG. 33 and FIG. 25 when the "T2 dout" signal and the "out" signal are gapped by a number of cycles. The gap is used as a reference to fix the data stream. It should be noted that the gap in cycles between these two signals is the same as the gap of buffering when the latch for T2 was waiting to insert its data.



Following the TRANSFORM in POSTC 233 part B, the interleaved stream is de-interleaved into "T2 out", as shown in FIGs. 18 and 23. The "T2 out" data stream has slip gaps in the data as described above. The T2 out [143:  $\phi$ ], shown in FIG. 17, enters a 16 to 1 multiplexor block, shown as block "DDPMUX" in FIG. 17. This multiplexor block will select data from one of 16 positions in the output buffer block, as shown in FIG. 25. This position is selected by the control logic, shown in FIG. 29, which uses the gap by which T2 "buffered-up" at its input. This gap is used as a reference. The output stream, T2DOUT, from the multiplexor block is the "fixed" data stream.

In range tests carried out on an embodiment of the present invention for the IDCT arrangement described above, it was found that all intermediate and final values were kept well within a known range at each point while still meeting the CCITT standards. Because of this, it was possible to "adjust" selected values as described above by small amounts (for example, by forcing certain bits of selected data words to desired values) without any fear of overflow or underflow in the arithmetic calculations.

The method and system, in accordance with the present invention, can be varied in numerous ways. For example, the structures used to resolve additions or multiplications may be altered using any known technology. Thus, it is possible to use resolving adders of subtractors where the preferred embodiment uses carry-save devices with separate resolving adders. Also, the preferred embodiment of the present invention uses down-scaling at various points to ensure that all values remain within their acceptable ranges. Down-scaling is not necessary, however, because other precautions may be taken to avoid overflow or underflow.

In one embodiment of the present invention, certain bits of various data words were manipulated to reduce the required word width within the system. However, the various intermediate values may, of course, be passed without bit manipulation. Furthermore, although only data words were bit-manipulated in the illustrated example of the present invention, it is also possible to manipulate the bits of constant coefficients as well and evaluate the results under the CCITT standard. If a comparison of the results showed that it would be advantageous to force a particular bit to a given value, in some cases, one might then be able to increase the number of "zeros" in the binary representation of these coefficients

in order to decrease further the silicon area required to implement the corresponding multiplier. Once again, bit manipulation is not necessary.

In summary of the above aspects of the present invention, the following is disclosed: an apparatus for transforming data having a first latch defining a first data stream source and a second latch defining a second data stream source. The first and second latches are in communication with a single arithmetic unit. The arithmetic unit communicates data to a transpose RAM, the transpose RAM transposes the data and communicates it to the second latch. The second latch is adjustable and can be varied in size to accommodate variable rates of data being received and transmitted. The second latch and first latch communicate 1st and 2nd data stream to the arithmetic unit sequentially, however, the sequential communication of the second latch does not interrupt the communication from the first latch. In this manner, common arithmetic unit is used for a first and second data stream. Furthermore, a process for transforming data using a common arithmetic unit having the following steps is described. First, loading the data into a first latch and, upon reaching a predefined number of cycles transmitting the data to an arithmetic unit and loading a first marker bit into a control shift register. Next, loading data into a second latch, the second latch is adjustable and can be varied in size to accommodate variable rate of data being received and transmitted at different rates. The next step is to transmit the data in the second latch to the arithmetic unit when the first control shift register reaches a predetermined state and the second latch is filled with a predetermined amount of data. Next, preventing transmission of data from the second latch, if the second latch is not filled with a predetermined amount of data and then recovering the second latch when the first latch is receiving non data.

# Detailed Description of Invention for Time Synchronization

In MPEG-2, video and audio data is synchronized using information carried in the MPEG-2 systems stream. In this regard, there are essentially two types of information that deal with synchronization; clock references and time stamps. Clock references are used to inform the decoder what number is used to represent the time "now". This is used to initialize a counter that is incremented at regular intervals so that the decoder always knows what the current time is.

Time stamps are carried in some of the streams of data that are used to make up the programme (typically video and audio). In the case of video, a time stamp is associated with a picture and tells the decoder at what "time" (defined by the counter that was initialized by the clock reference) a picture should be displayed.

In MPEG, multiplexed into the system stream are a series of clock references. These clock references define the "system time". There are two types of clock reference; Program Clock References (PCRs) and System Clock References (SCRs). In the present invention, the distinction between PCRs and SCRs is not relevant since each of the clock references are used in the same manner by the decoder. PCRs and SCRs have timing information to a resolution of 90 kHz with a further field extending the resolution to 27 MHz (or  $1/27 \times 10^6$  in seconds). Clock references are included in the data stream fairly often in order that "system time" may be reinitialized after a random access or channel change. Accordingly, it is important to appreciate that timestamps refer to a hypothetical model of a decoder that can decode pictures instantly. As will be appreciated by one of ordinary skill in the art, any real decoder cannot do this and must take steps to modify the theoretical time in which pictures should be displayed. Furthermore, time stamps and the clock references are used to determine display time and errors in display time. This modification depends upon the details of the architecture of the particular decoder. Clearly any delay introduced by the video decoder must be matched by an equivalent delay in the audio decoder.

When decoding MPEG, discontinuities in the concept of "system time" may occur. For instance in an edited bitstream, each edit point will have discontinuous time. A similar situation occurs at channel change. It will be appreciated that care must be taken when using time stamps, because using a time stamp

that was encoded in one time regime with respect to a "system time" defined by a clock reference from another regime will clearly lead to incorrect results.

Figure 39 shows the demultiplexing of the MPEG systems stream into *elementary streams* 250. Each elementary stream will typically carries either video or audio data although, in general, any form of data may be transported. Each elementary stream is divided into a series of *access units*. In the case of video, the access unit is a picture. In the case of audio, it is a fixed number of samples of audio data.

Also multiplexed into the systems stream are a series of clock references.

These clock references define the "system time".

In accordance with the present invention, associated with each elementary stream is a series of time stamps 251. The time stamps specify the "system time" at which the next access unit for the respective elementary stream is to be presented. These time stamps are referred to as *presentation time stamps*, "PTS".

In the case of video data, a second type of time stamp is also defined is referred to as a *decode time stamp*, "DTS". Since the DTS is only present when a PTS is also present and there is a simple relationship between them, the detailed differences between these two types of timestamps can be ignored since PTS/DTS differences have no bearing on the present invention.

The decode time stamps (DTS) define the time at which an access unit (picture in the case of video) is to be decoded. The presentation time stamps (PTS) define the time at which an access unit is to be presented (displayed). However, the timing model used is a hypothetical model in which the decoder is infinitely fast. In this case, the DTS and PTS would be identical to one another.

However, in MPEG video decoding, some of the pictures are reordered. Therefore, after decoding, the pictures are held in storage for a time period, e.g., several frame times, before they are displayed. During this time period, other pictures that are decoded subsequent to the picture in question are displayed. Consequently, for these reordered pictures there is a difference between the DTS and PTS.

In accordance with the present invention, it will be appreciated that to properly synchronize time, it is necessary to be consistent in the use of time stamps. In one preferred embodiment, the time synchronizing circuitry is placed

at a point in the decoding pipeline when the pictures occur in their decoded order. Accordingly, this embodiment uses the OTS.

Nevertheless, the circuitry could equally be moved to a point in the decoding pipeline that occurs after the pictures are reordered and, therefore, the pictures would reach the synchronizing circuitry in their display order. Hence, as  
5 will be appreciated by one of ordinary skill in the art, PTS would be used in this embodiment.

In the preferred embodiments of the present invention, the information derived from the timestamps is transported through the various circuits by means of tokens. Tokens consist of a series of one or more words of information. The first word of the token contains a code which identifies the type of token and, hence, the type of information carried by that token. Associated with each word of the token is an extension bit which is set to one to indicate that there are more words in the current token. Therefore, the last word of a token is indicated by  
10 the extension bit being zero. In the present invention, the code in the first word indicating the type of token may be of a variable number of bits so that some codes use a small number of bits (allowing the remainder of the bits in the first word to be used to represent other information) while other codes use a larger number of bits.

Tokens may be characterized as being either control or DATA tokens. For example, at the interface between the system decoder and the video decoder, there are two types of information: (1) the coded video data and (2) the synchronization time derived from the time stamp information. The coded video data is viewed as data and is carried in a DATA token (e.g., the token called DATA)  
15 while the synchronization time is viewed as control information and is carried in a control token (called SYNC\_TIME). Additional control tokens may also be used from time to time in the present invention. For example, a FLUSH token that behaves in a manner similar to a reset signal may be required to initialize the video decoding circuitry before attempting to restart decoding because of an  
20 error.

In accordance with the present invention, it is an object of one preferred embodiment to time synchronize two circuits and, more particularly, to time synchronize two circuits without directly communicating system time from the first to the second circuit. In accordance with the invention, time synchronization of

two circuits is accomplished without passing system time directly to the second circuit by providing synchronized time counters in each circuit.

The present invention also enables the system to time synchronize two circuits without communicating system time from the first to the second circuit by providing an elementary stream time counter in each circuit.

Accordingly, another object of the present invention is to time synchronize two circuits and to determine the presentation time error, if any, of the object being presented by using time stamp information, system time, and elementary stream time from the first circuit to generate synchronization time passed to the second circuit and compared to a copy of elementary stream time in the second circuit which is synchronized with the elementary stream time in the first circuit. The system of the present invention can time synchronize a system decoder and a video decoder without directly communicating system time from the system decoder to the video decoder, without passing system time directly to the video decoder by providing synchronized time counters in each circuit and without communicating system time from the system decoder to the video decoder by providing a video counter in each circuit.

The invention also enables the system to time synchronize a system decoder and a video decoder and to determine the display time error, if any, of the picture being displayed by using video time stamp information, system time, and video decoding time from the system decoder to generate synchronization time which is then passed to the video decoder and compared to a copy of video decoding time in the video decoder which is synchronized with the video decoding time in the system decoder.

In accordance with the present invention, information derived from the timestamps can be transported through the system using a control token as previously described.

Figure 40 shows a first preferred embodiment implementing elementary stream timestamp management, in accordance with the present invention. The clock references 253, which represent system time, are decoded by the system demultiplexer 254 and placed initially, and then as needed, into a time counter 255 within the system decoder 256, and are incremented at 90 kHz. A second copy of the clock reference 253 is simultaneously loaded into the time counter 255 that is inside the elementary stream decoder 257, incremented also at 90 kHz, and synchronized to the time counter 255 in the system decoder 256.

75

The time stamps 251, in accordance with the present invention, flow from the system demux 254 through the elementary stream buffer 260 so that they are delayed by the same amount as the incoming data. The time stamps 251 may also have a correction added to compensate for the non-zero decode time of the elementary stream decoder 257. The corrected time stamps 251 are then compared with the copy of the time stored the time counter 258 inside the elementary stream decoder 257 to determine whether the decoded information is presented too early or too late.

The above embodiment is better than merely passing system time directly to the elementary stream decoder 257 from the time counter 255 in the system decoder 256 because the counter in the system decoder changes 90,000 times a second. Therefore, system time would, in all essence, need to be continually passed to the elementary stream decoder 257. Passing system time continually would require dedicated pins or the like. By using a time counter 255 located in the system decoder 256 and a time counter 258 located in the elementary stream decoder 257, system time can be passed in the form of clock references 253 a few times a second.

Another embodiment is shown in Figure 41. The embodiment shown in Figure 41 avoids the need for the clock references 253 to be passed to the elementary stream decoder 257. This is achieved by using a second counter "es\_time" 262, containing information on elementary stream time, which is maintained in both the system decoder 256 and the elementary stream decoder 257. The two es\_time counters 262 and 263 are reset at power on, and at other times such as channel change, and then they free run from there on. Since this embodiment depends on the two es\_time counters 262 and 263 staying in step, it will be appreciated that it is necessary to take measures to ensure the es\_time counters do not get out of step. One way to ensure the es\_time counters 262 and 263 stay in step is to use carry out of the es\_time counter in the system decoder to reset the es\_time counter in the elementary stream decoder 257 as shown in Figure 41.

As further shown in Figure 41, the clock references 253, which represent system time, are decoded by the system demultiplexer 254 and placed into a time counter 255 within the system decoder 256 and incremented at 90 kHz. The es\_time counter 262 in the system decoder 256 of the present invention and the es\_time counter 263 in the elementary stream decoder 257 of the present

invention are synchronized with each other and incremented at 90 kHz. Elementary stream time stamps are also decoded by the system demultiplexer 254. Accordingly, a synchronization value X is computed using the elementary stream timestamp, the system time contained in the time counter and the elementary stream time contained in the es\_time counter 262 contained in the system decoder 256 according to the equations 3-1.

The following set of equations 3-1 (a - d) is illustrative of one method in accordance with the present invention, for time synchronization which avoids passing the clock references 253 to the elementary stream decoder 257.

Equation 3-1 (a) is the equation required for time synchronization. Since it is undesirable to pass system time directly to the elementary stream decoder circuit 257, as shown in Figure 41, a synchronization time representation X is generated, using Equation 3-1 (b-d), by the system decoder 256 and this value is passed to the elementary stream decoder. Synchronization time X is then compared to the elementary stream time contained within the es\_time counter 263 located within the elementary stream decoder 257. Hence, the compared result is used to determine whether the decoded information is presented too early or too late and then is further used in time synchronizing the system.



Equations 3-1:

- a) Time Synchronization = (Elementary stream timestamp - system time)
- b) Time Synchronization = (X - elementary stream time)
- c) (X - elementary stream time) = (elementary stream timestamp - system time)
- d)  $X = (\text{elementary stream timestamp} - \text{system time} + \text{elementary stream time})$

In the present invention, the synchronization time, X, may have a correction added to compensate for the non-zero decode time of the elementary stream decoder 257. The corrected synchronization time is then compared with the elementary stream time contained in the es\_time counter 263 located inside the elementary stream decoder 257 to determine whether the decoded information is presented too early or too late and is further used to time synchronize the system. Note, the time correction could be subtracted from elementary stream time contained in the es\_time counter 263 located inside the elementary stream decoder 257 instead of added to synchronization time X for the same result. The above embodiment is an example of a solution for generating synchronization time X and determining whether the information is presented early or late. It will be apparent to those skilled in the art that there are many other equivalent solutions for accomplishing the above.

For example, Figure 42 shows an alternative method for determining the synchronization time, X, in accordance with the present invention. In this arrangement, the system decoder 256 does not maintain an elementary stream time. Instead it records, in the register initial\_time 265, the value of system time at the instant that the elementary stream time counter, es\_time 263, located in the elementary stream decoder 257 is reset to zero. The value in es\_time 263 can be computed by the system decoder 256 because it will be the difference between the current system time and the value recorded in initial\_time.

The following equations 3-2 (a-c) is illustrative of this alternative method for time synchronization. Equation 3-2 (a) is the equation representing the value of the elementary stream time stored in es\_time 263 located in the elementary stream decoder 257. This is substituted into equation 3-1 (d) to give equation 3-2 (b) which is simplified to derive equation 3-2 (c) providing the synchronization time, X, as a function of the system time and the value stored in the initial\_time register 265.

Equations 3-2:

- a) elementary stream time = system time - initial\_time
  - b)  $X = (\text{elementary stream timestamp} - \text{system time} + [\text{system time} - \text{initial\_time}])$
  - 5 c)  $X = (\text{elementary stream timestamp} - \text{initial\_time})$
- Two solutions for deriving the synchronization time,  $X$ , in accordance with the present invention have been illustrated. However, it will be apparent to those skilled in the art that there are many other equivalent solutions.

Figure 43 shows another embodiment of the present invention implementing video timestamp management. The clock references 253, which represent system time, are decoded by the system demultiplexer 254 and placed initially, and then as needed, into a time counter 255 within the system decoder 256 and are incremented at 90 kHz. A second copy of the clock references 253 are simultaneously loaded into the time counter 258 that is inside the video decoder 270 and incremented at 90 kHz, and synchronized to the time counter 255 in the system decoder 256.

The video time stamps flow from the system demux 254 through the video decoding buffer 271 so that they are delayed by the same amount as the incoming video data. The video time stamps may have a correction added to compensate for the non-zero decode time of the video decoder 270. The corrected video time stamps are then compared with the copy of the time in the time counter 258 inside the video decoder 270 to determine whether the decoded picture is displayed too early or too late.

The embodiment shown in Figure 43 is an improvement over the process of merely passing system time directly to the video decoder from the time counter in the system decoder because the counter in the system decoder changes 90,000 times a second. Therefore, system time would in all essence need to be continually passed to the video decoder. Passing system time continually would require dedicated pins or the like. By using a time counter located in the system decoder and a time counter located in the video decoder system time can be passed in the form of clock references a few times a second.

Referring now to Figure 44, the clock references, which represent system time, are decoded by the system demultiplexer 254 and placed into a time counter 255 within the system decoder 256 and incremented at 90 kHz. The vid\_time counter 272 in the system decoder 256 and the vid\_time counter 273 in

the video decoder 270 are synchronized with each other and incremented at 60 kHz. Video time stamps are also decoded by the system demultiplexer 254. Accordingly, a synchronization value X is computed using a video timestamp, the system time contained in the time counter 273 and the video decoding time contained in the vid\_time counter 272 contained in the system decoder 256 according to the equations 3-3.

The following set of equations 3-3 (a - d) is illustrative of one method in accordance with the present invention, for time synchronization which avoids passing the clock reference 253 to the video decoder 270. Equation 3-3(a) is the equation required for time synchronization. Since it is undesirable to pass system time directly to the video decoder circuit 270 as shown in Figure 44, a synchronization time representation X is generated, using Equation 3-3 (b - d), by the system decoder 256 and passed to the video decoder 270. Synchronization time, X, is then compared to the video decoding time contained within the vid\_time counter 273 located within the video decoder 270. The compared result is used to determine whether the decoded picture is displayed too early or too late and then further used in time synchronizing the system.

Equations 3-3:

- a) Time Synchronization = (Video timestamp - system time)
- b) Time Synchronization = (X - video decoding time)
- c) (X - video decoding time) = (video timestamp - system time)
- d)  $X = (\text{video timestamp} - \text{system time} + \text{video decoding time})$

In the present invention, the synchronization time, X, may have a correction added to compensate for the non-zero decode time of the video decoder. The corrected synchronization time is then compared with the video decoding time contained in the vid\_time counter 273 located inside the video decoder 270 to determine whether the decoded picture is displayed too early or too late and is also used to time synchronize the system. Note, the time correction can be subtracted from the video decoding time contained in the vid\_time counter 273 located inside the video decoder 270 instead of added to synchronization time X for the same result. The above embodiment of the present invention is another example of a solution for generating synchronization time X and determining whether the picture is displayed early or late. However, it will be apparent to those skilled in the art that there are many other equivalent solutions for accomplishing the above.

Another nice feature, in accordance with the present invention, is that there is no need to deal with the full 33 bit time stamp number or 42 bit clock reference number. The present invention restricts the counters to 16 bits to allow 16 bit handling on the video decoder 270. At first glance, it would appear that 16 bits cannot represent a sufficient number range at a resolution of 90 kHz (only 2/3 second to be used). However, there is no need for such high precision because the time control on the video decoder 270 is only accurate to a field time (since the video timing generator VTG free-runs or is gen-locked to something that has nothing to do with the MPEG stream being decoded) and, therefore, it is not related to timestamps or presentation time in any way.

As shown in Figure 44 and Figure 45, the synchronization time X and the vid\_time counter 273 within the video decoder 270 use only sixteen bits. This is made possible by two factors. First, the difference between system time and the timestamp (used to derive the synchronization time, see Equation 3-3) should always be small, thus allowing the more significant bits to be discarded. Second, it is only possible to control the presentation of video to the nearest frame time. Therefore, the less significant bits are not required and are discarded by shifting right by four bits.

Thus, the sixteen bits of time information used in the present invention are able to deal with timing errors of up to about 11.5 seconds with an accuracy of about 180  $\mu$ s (about 1% of a field time). A PAL or SECAM European 625 line TV system is, thus, 112.5 ticks of the 5625 Hz clock; a NTSC 525 line TV system is 93.84 ticks. Hence, using 16 bits allows timing calculations with an accuracy of about 1% of a field time.

Figure 46 shows the preferred process, in accordance with the present invention, of the moving the time stamp through the hardware. The preferred method for communicating information in this hardware is Tokens, but it will be appreciated that alternative methods may also be employed. The hardware is divided into two modules. The first module is added just after the Start Code Detector 201. This module is responsible for generating a token, SYNC\_TIME containing the synchronization time X, as discussed above, and this occurs just before an associated PICTURE\_START token. In the MPEG systems stream, the time stamp is carried in a packet header and refers to the first picture in the packet of data. Since the packets do not line up with the video data, there will, in

general, be the end of the previous picture before the start of the picture to which the time stamp refers.

The synchronization time information may be supplied to the present invention either via a microprocessor interface or by using a Token. In either case, the synchronization time date (16 bits) is stored in the synchronization time register (divided into two parts to allow access to each byte individually), as further detailed in Table 12.

5

10

Register	Size/Dt	Reset	Description
ts_low	8/bw	-	<p>The lower eight bits of the synchronization time value.</p> <p>The ts_low register is slaved so that new values may be written into this register without affecting the value previously written (that will become part of a SYNC_TIME token).</p> <p>Writes to ts_low register affect the master register whilst reads read-back the slave register. Until a master-to-slave transfer has been effected using ts_valid the value written into ts_low cannot be read back.</p>
ts_high	8/bw	-	<p>The upper eight bits of the synchronization time value.</p> <p>Slaved in the same way as ts_low.</p>

12

Register	Size/Dt	Reset	Description
ts_valid	1bw	0	<p>This bit controls the master-slave transfer of ts_low and ts_high.</p> <p>When values have been written into ts_low and ts_high the microprocessor should write the value one into this bit. It should then poll the bit until it reads back the value one. At this point the values written into ts_low and ts_high will have been transferred into the slave registers (and can be read back) and ts_waiting will be set to one.</p> <p>The microprocessor should then write the value zero in preparation for the next access.</p>
ts_waiting	1bw	0	<p>When set to zero the registers ts_low and ts_high do not contain valid synchronization time information.</p> <p>When set to one the registers ts_low and ts_high contain valid synchronization time information. A SYNC_TIME token will be generated before the next PICTURE_START token and ts_waiting will then become zero.</p> <p>This bit should be polled to ensure that it is zero before writing a one into ts_valid to ensure that the previous synchronization time value has been used before it is overwritten by the master-to-slave transfer.</p>

Table 12 Microprocessor registers for handling synchronization time

5

In the present invention, a flag, ts\_waiting, is set to indicate the fact that valid synchronization time information is in the timestamp register. If the data was

supplied using the SYNC\_TIME token, then that token is removed from the stream of tokens.

When a PICTURE\_START token is encountered, the flag that indicates the status of the synchronization time register is examined. If the flag is not set, then no action is taken and the PICTURE\_START token and all subsequent data is unaffected. If, however, the flag is set, indicating that valid synchronization time information is available in the register, then a SYNC\_TIME token is generated and placed in the data stream before the PICTURE\_START token. The flag is then cleared and the synchronization time register is made available for the next time-stamp that occurs.

The second module as shown in Figure 46, consists of a prescaler clocked at 27 MHz and a vld\_time counter clocked by the prescaler 278 which are associated with the microprogrammable state machine, (MSM) 218.

There is a prescaler 278 that divides the clock by 4800, as shown in Figure 44 and Figure 45. In other words, 4800 is 300 (27 MHz / 90 kHz) times 16. The 4804.8 option shown in Figure 45 and Figure 46 is discussed below.

In the NTSC color television, the frame rate is not 30 Hz but is, in fact, approximately 29.94 Hz (precisely 30000/1001 Hz). [Before the advent of color television 30 Hz precisely was used.] There are precisely 1716, 27 Mhz clock periods per NTSC line time (line time is 1/525 of frame time).

The American television standards body has expressed an interest in returning to 30 Hz in the future (or more probably 60 Hz for HDTV). As a result MPEG supports a frame rate of 30 Hz precisely. However, since it is not possible to generate a stable 30 Hz television signal from a 27 MHz clock (there being 1714.29...cycles per line) it is difficult to generate a 30 Hz raster in the MPEG framework.

One possible solution is to "bend" the clock rate at the decoder so that instead of producing a 27 MHz clock, a 27.027 MHz clock is generated. This clock is generated using the MPEG clock references with a divider of 300.3 (rather than 300) to yield the 90 kHz clock. This 27.027 MHz clock when clocking the identical video timing circuitry that provides a 29.94 Hz frame rate from 27 MHz will give a precise 30 Hz rate.

In the framework of the present invention, the 90 kHz is prescaled by a further factor of 16. Accordingly, division of the 27.027 MHz clock by 300.3x16=4804.8.

The Vid\_time counter 273 (discussed above) contains the video decoding time and is incremented each time that the prescaler reaches its terminal count. The vid\_time counter 273 is reset by the reset-time pin.

5 The prescaler and vid\_time counter of the present invention can be implemented with fully clocked feed-back flip-flops which are much more resistant to  $\alpha$ -particle corruption than the resistive-feedback or weak-feedback latches used elsewhere. Using clocked feedback flip-flops for time counters will help ensure that the time counter in the video decoder stays in step with the time counter in the system decoder.

10 Figure 47 illustrates the process the MSM 218 performs when it receives the SYNC\_TIME token. The MSM 218 is able to read the current time indicated by the video time counter and to then compare it with the value supplied by the video SYNC\_TIME token. It can, therefore, determine whether it is early or late, as compared to the time at which it should be decoding the pictures.

15 In the present invention, a 16 bit signed timestamp correction is added to the synchronization time X (discussed above) that was carried by the video SYNC\_TIME token. This correction is reset to zero by the MSM 218 at chip-reset, and if no action is taken, the synchronization time remains be unaltered. The controlling microprocessor can always write value into ts\_correction to modify the synchronization time and, therefore, compensate for differential delays through the video and audio decoders.

20 The current video decoding time contained in the vid\_time counter 273 is subtracted from the corrected synchronization time. The sign of value gives the direction of the error (and determines the error code, if any, generated by the MSM 218). The absolute value of the difference is then taken and the result is compared to a threshold value to determine whether the timing error is within acceptable limits. Since, at present, the video timing can only be controlled to an accuracy of plus or minus a frame time from the nominal time (because the VTG 333 free-runs) this threshold is set at one frame time.

25 If the error exceeds a frame-time, then some correction must be made. The MSM 218 of the present invention can correct the situation itself if the decoding is too early since the MSM can simply delay the decoding until the appropriate time. However, if the decoding is later than the intended time, then time correction is more difficult because it is not possible to discard pictures reliably at the output of the coded data buffer. Essentially, the decoding of the

30

35



sequence is broken and the most reliable way to correct the situation is to restart the decoding process in a manner similar to random-access or channel change.

- In order to facilitate this process, the control register of the MSM 218 may be programmed to discard all data until a suitable start code or FLUSH token is encountered. In addition, the error "ERR\_TOO\_EARLY" is not generated during start-up, irrespective of the setting of disable\_too\_early, because at start up, the first picture is expected to be early.

Table 13 is illustrative of how the MSM 218 registers work and details some of the actions and error messages information placed in the registers can generate.

Register Name	Size/Dir	Reset State	Description
ts_correction	16/rw	zero	Correction added to synchronization time before it is used.
frame_time	16/rw	226 or 188	Represents the tolerance on the timing of decoding pictures. Reset state determined by the PALNTSC pin.
vid_time	16/ro	zero	Reset by either reset or reset_time. The current value of video decoding time.
manual_startup	1/rw	zero	When set to one the start-up is to be performed manually using decode_disable. In this case SEQUENCE_END and FLUSH tokens at the MSM cause decode_disable to be set to one.

Register Name	Size/Dir	Reset State	Description
decode_disable	1/rw	zero	<p>When set to zero the decoding proceeds normally.</p> <p>At the start of each picture the MSM checks the status of decode_disable and will not proceed if it is set to one.</p> <p>Note that if manual start-up is to be performed (i.e. without the time-stamp management hardware) then this bit should be set to one at the same time as manual_startup is set to one.</p>
disable_too_early	1/rw	zero	<p>When set to one the error "ERR_TOO_EARLY" indicating that the decoding is too early is suppressed and the MSM simply waits to correct the situation.</p>
NTSC_30	1/rw	zero	<p>When set to one the prescaler divides by 4804.8 rather than 4800. Set automatically when decoding 30 Hz frame rates.</p>
discard_if_late	1/rw	zero	<p>This has no effect unless an "ERR_TOO_LATE" is generated (or would be generated if errors were not masked out). If it is set to one then data is discarded until the condition indicated by discard_until.</p>

Register name	Size/Direction	Reset State	Description
discard_until	2/rw	zero	<p>Indicate the condition which causes time-stamp triggered discarding to be terminated.</p> <p>0 - FLUSH  1 - SEQUENCE_START  2 - GROUP_START  3 - NEXT PICTURE</p> <p>Note 1 - that discarding one picture may immediately be un-done if that picture is a field picture by the generation of a dummy field to preserve the alternating top/bottom field structure. As a result if discard_until is set to "Next Picture" but the dummy field would be generated one further picture is discarded.</p>

Table 13 Timestamp MSM registers

As a result of the synchronization time handling of the present invention, it is possible that one of two errors will be generated.

ERR\_TOO\_EARLY is generated if the decoding is taking place earlier than the time indicated by the time-stamp. ERR\_TOO\_EARLY may be suppressed, but

ERR\_TOO\_LATE will always be generated unless all errors are masked out.

In summary, the present invention includes: an apparatus for synchronizing time having, a timestamp for determining presentation time, a clock reference for initializing system time in a first circuit, a first time counter in communication with the clock reference for keeping system time in a first circuit and a second time counter initialized by the clock reference in a second circuit synchronized with the first time counter, for keeping a local copy of the system time and for determining the presentation timing error between the local copy of system time and system time by

comparing the timestamp to the second time counter. It further includes an apparatus for synchronizing a system decoder and a video decoder using a timestamp for determining display time, a clock reference for initializing system time in the system decoder, a first time counter in communication with the clock  
5 reference for keeping system time in the system decoder and a second time counter initialized by the clock reference in the video decoder synchronized with the first time counter, for keeping a local copy of system time and for determining the display timing error between the local copy of system time and system time by comparing the timestamp to the second time counter. A still another embodiment includes an  
10 apparatus for synchronizing a first circuit and a second circuit using a clock reference for initializing system time in the first circuit, a first circuit having a time counter in communication with the clock reference for keeping system time, a first elementary stream time counter in the first circuit for providing elementary stream time. The first circuit is adapted to receive a time stamp, and the first circuit  
15 generates synchronization time by adding elementary stream time to the time stamp and subtracting system time. The second circuit is adapted to receive synchronization time from the first circuit and has a second elementary stream time counter in synchronization with the first elementary stream time counter for providing a local copy of the elementary stream time and for determining a timing error  
20 between the system time and the time stamp by comparing synchronization time to the local copy of elementary stream time. In this way, the clock reference signal does not have to be passed directly to the second circuit in order to determine the timing error. In another embodiment, an apparatus for synchronizing a first circuit and a second circuit has a clock reference for initializing system time in the first  
25 circuit. The first circuit has a time counter in communication with the clock reference for keeping system time, and a first video time counter for providing video decoding time. The first circuit is adapted to receive a video time stamp and generates synchronization time by adding video decoding time to the video time stamp and subtracting system time. The second circuit is adapted to receive  
30 synchronization time from the first circuit and has a second video time counter in synchronization with the first video time counter for providing a local copy of video decoding time and for determining a timing error between system time and the video time stamp by comparing synchronization time to the local copy of video decoding time. Accordingly, the clock reference signal does not have to be passed directly  
35 to the second circuit in order to determine the timing error. The present invention

also includes a method for providing timing information by providing a video data stream having a time stamp carried in packet header wherein the time stamp refers to the first picture in the packet of data. In the next step a register is provided having a flag used to indicate valid time stamp information which is taken from the packet header and placed into the register. Next, the timestamp is removed from the video data stream and placed in the register. Next, the method encounters a picture start and subsequently examines the status of the register to determine if valid time stamp information is contained in the register by checking the flag status. A time stamp is generated in response to the picture start if the flag indicates valid time stamp information is contained in the register and then the timestamp is inserted back into the data stream. Another embodiment includes an apparatus described above wherein the elementary stream time counters are restricted to 16 bits. Likewise, there is an apparatus as described above, wherein the second elementary stream time counter located in the elementary stream decoder is restricted to 16 bits. Furthermore, there is an apparatus as described above wherein the synchronization time is restricted to 16 bits for controlling the elementary stream decode. The present invention also has a process for decoding video and for determining display time errors against a threshold value. It then parses video data into tokens for further processing, determining if a time stamp token is indicated, comparing the time stamp token to a video time, and generates a compared value to determine an indicative of timing error. Next, it determines whether the compared value, when compared against a threshold value, is within acceptable parameters when a timing error is indicated and indicates when the compared value is outside acceptable parameters. An alternative embodiment includes an apparatus for using a system decoder and a video decoder. The system decoder is adapted to accept MPEG system streams and demultiplexing video data and the video time stamp from the stream. The system decoder has a first time counter representative of system time. The video decoder accepts the video data and the video time stamp, and has a second time counter in synchronization with the first time counter. The video decoder also has a video decoder buffer for accepting the video data at a substantially constant rate and outputting the video data at a varying rate and for passing a video time stamp. The video decoder while decoding a picture from the video data also compares the video time stamp for the decoded picture with the second time counter to determine the appropriate display time. There is also a method for determining a time error between a first

circuit and a second circuit by providing the first circuit with a system time (SY), a time stamp (TS), and an elementary stream time (ET), obtaining synchronization time (X) by using the elementary stream time (ET), the time stamp (TS) and the system time (SY), in accordance with the equation:  $X = ET + TS - SY$ , providing  
5 synchronization time (X) to the second circuit and generating a synchronized elementary stream time (ET2) and obtaining a time error by using synchronized time (X) and in accordance with the equation  $ET2 - X$ ; hence, the first circuit can be time synchronized with the second circuit without passing system time to the second circuit. Another method for determining a time error between a first circuit and a  
10 second circuit has the following steps: providing the first circuit with a time stamp (TS), and an initial time (IT), obtaining synchronization time (X) by using the time stamp (TS) and the initial time (IT), in accordance with the equation  $X = TS - I$ , providing synchronization time (X) to the second circuit and generating a synchronized elementary stream time (ET) and obtaining a time error by using synchronized  
15 time (X) and in accordance with the equation  $ET - X$ . In this way, the first circuit can be time synchronized with the second circuit without passing system time to the second circuit. Still another method for determining a time error between a first circuit and a second circuit includes the following steps: providing the first circuit with a system time (SY), a video time stamp (VTS), and a video decoding time (VT),  
20 obtaining synchronization time (X) by using the video decoding time (VT), the video time stamp (VTS) and the system time (SY), in accordance with the equation:  $X = VT + VTS - SY$ , providing synchronization time (X) to the second circuit and generating a video decoding time (VT2) in the second circuit which is synchronized to the video decoding time (VT) in the first circuit, and obtaining a time error by using  
25 synchronized time (X) and in accordance with the equation  $VT2 - X$ . Accordingly, the first circuit can be time synchronized with the second circuit without passing system time to the second circuit.

#### Detailed Description of the Invention for Asynchronous Swing Buffering

For asynchronous swing buffering, in accordance with the present invention, two buffers are operated asynchronously; one is written while the other is read. Accordingly, this allows for a data stream having a first rate of through-put to be resynchronized to another rate, while still maintaining a desired rate. In the invention, the write control and read control both have state indicators for communicating which buffer they are using and whether the controls are waiting for access or are, in fact, accessing that buffer. Each side communicates to the other side a single bit to indicate which buffer it is using. This is the only signal that must be synchronized between the two sides of asynchronous circuitry.

When one control circuit (read or write) finishes accessing a buffer, then the invention will allow control to pass to the other circuit. If, after the control has swung, and two control circuits are trying to use the same buffer, then the later control circuit will begin waiting. The control circuit will wait until each side is using alternate buffers, i.e., the other side has swung. If, after it has swung, it finds that it is now using the alternate buffer to the other side, it will not wait, but immediately commence accessing. This system of arbitration between the buffers is started up by both buffers using the same buffer, buffer 0, in this case. The read side starts up by waiting, while the write side is accessing, since there is nothing valid to read out of either buffer.

In one embodiment, in accordance with the present invention, the swing buffers are two discrete RAMs having all signals, such as enabling strobes, addresses and data multiplexed from either the read or write side, dependent on which buffer is being accessed by each side. This structure has been shown to use a lot of area in the busing of a large number of signals between the two buffers.

Combining the two RAMs into a single structure saves much of the busing area while still maintaining performance to the same standard. This structure contains twice as many rows of cells as one of the discrete RAMs found in the first embodiment of the present invention. However, the second embodiment must have two pairs of bit lines since the read and write to the discrete buffers is happening simultaneously and asynchronously. Each row will be of its original width (i.e., have the same number of cells) since accesses are the same width as for the discrete RAMs. Each pair of rows are accessed as if at the same address, but from different buffers, so they connect to a different pair of bitlines. Using the same address, these pair of rows can be readily accessed by one row decoder connected to the

read address and one row decoder connected to the write address. Again, the read and write control never access the same buffer at the same time so there is no conflict as to which pair is accessed by which row decoder.

In the same way in which each row decoder can access rows from each buffer, both the read and write circuitry within the structure of the present invention connect to each pair of bitlines, one pair from each buffer. The read and writes are then multiplexed into each of the buffers and, for the same reasons explained above, there will not be conflict.

As shown in Figure 48, a swing unit 1 includes swing buffers 10 with RAM 12 and 14 in accordance with the present invention. The swing unit 1 also includes a write control circuit and a read control circuit, which control the data into and out of the RAM 12 and 14. The read control circuit and the write control circuit accomplish this by use of strobes, data and address control lines, 8. Lines 7 and 9 are control lines to indicate the RAM used by the write control circuit and the RAM used by the read control circuit. Line 7 functions to control the write control circuitry, i.e., when the read control circuitry is using, RAM 12 if low, RAM 14 if high. Similarly, Line 9 functions to inform the read control circuitry that the write control circuitry is using RAM 12 if low, RAM 14 if high.

In the present invention, swing buffer 10 has two RAM arrays, 12 and 14. Swing Buffer 10 is capable of asynchronous, alternative reading and writing to the RAM area which enables the apparatus to achieve the necessary band width for high speed accessing of the memory. The RAMs 12 and 14 require the following signals: write address 16, read address 18, data in 20, data out 22; and a read and write enable signal (not shown). See also Figure 49.

The write address and read address signals are multiplexed by multiplexers 24. The RAM array 12 and 14 operate with the write circuitry, row decoder and read circuitry in a conventional sense.

In the first embodiment of the present invention, during initiation of the swing buffer 10, RAM 12 will be written to until the control circuitry switches a write enable single to RAM 14.

Once the RAM array 12 has been written, it falls under the control of the read circuitry 4, to be read. During this time, the RAM array 14 is also being written. It is important to note when the RAM array falls under the control of the read array control 2, or the write control circuit 4, the control is established until reading or writing is completed and then control is turned over. In the situation where the read



control circuit 4 is accessing the RAM array, such as 12, and the write control circuitry 2 needs to access the same RAM array 12, then the write control circuit will begin waiting.

Therefore, in accordance with the present invention, two control events are created. When a write control circuit or a read control circuit swings to a different RAM, it will either begin immediately accessing the RAM since the RAM is free and not under control of the alternative circuit, or it will begin to wait. During start up, the read side defers to the write side, since there is nothing valid to be read out of either buffer.

The second embodiment of the present invention is shown in Figure 50. An integrated swing buffer 30 includes a RAM array 32 having the logical size of RAM array 12 combined with RAM array 14. In other words, there is the same amount of RAM in both the first and second embodiments, however, it is combined in the second embodiment. Accordingly, the integrated swing buffer has the advantage of saving much of the busing area while still performing the same swing buffer function.

In the second embodiment of the present invention, the write circuit and read circuit 34 and 36 respectively, are similar to those used in the swing buffer 10. However, these circuits now include selectors which choose from the pairs of bit lines described hereinafter. Likewise, the read access row decoder 38 and the write access row decoder 40 are similar to those contained in swing buffer 10, however, they are able to access a pair of rows as described hereinafter in Figure 51.

As shown in Figure 51, the particular structure of the integrated swing buffer 30, in accordance with the present invention, is detailed. Individual cells 42 are contained in rows 44. The read row decoder 38 and write row decoder 40 access the rows 44 in pairs. A pair of rows have the same address provided by the address lines 16 and 18. The read buffer line 52 and write buffer line 54 provide the control information for selecting one of the paired rows 42. The buffer 0 bitlines 48 and buffer 1 bitlines 50 connect to alternative rows of cells and to the read and write circuitry 34 and 36. For clarity in depicting the addressing, the lighter shading illustrates the read row decoder 38 accessing a row in buffer 0. Similarly, the darker shading illustrates the write row decoder 40 accessing a row in buffer 1.

In summary, the present invention includes a swing buffer apparatus having at least two RAM arrays, a write control circuit in communication with the RAM arrays for controlling data input into the RAM array, and a read control circuit in

- communication with the RAM arrays for controlling data output from the RAM arrays. Furthermore, the write control circuit and read control circuit are in communication with one another to allow a synchronized control of the RAM arrays. There is also a swing buffer apparatus having a RAM array, a write control circuit in communication with the RAM array through a pair of bit lines, a read control circuit in communication with the RAM array through another pair of bit lines and a read row decoder and a write row decoder for addressing the RAM through a pair of rows so that individual cells are read. The present invention also provides a method of asynchronously addressing RAM, by decoding at least a pair of cells in the RAM, using a row decoder to decode at least a pair of rows and selecting one of the rows to be assessed, using at least two pairs of bitlines connected to read a circuit and a write circuit and selecting the pair of bitlines to be used.

# DETAILED DESCRIPTION OF THE INVENTION FOR STORING VIDEO INFORMATION

Video decompression systems contains three basic parts used to decode and display picture information. The three main parts of a video decompression system are the spatial decoder, temporal decoder and the video formatter. The present invention involves the temporal decoder and video formatter and the way in which the temporal decoder and video formatter manage their respective picture buffers, hereinafter the frame store buffer. In MPEG systems, the temporal decoder contains two frame store buffers and the video formatter contains two frame store buffers.

MPEG uses three different picture types: Intra (I), Predicted (P) and Bidirectionally interpolated (B). B pictures are based on predictions from two other pictures; one picture is from the future and one from the past. The I pictures require no further decoding by the temporal decoder, but must be stored in one of the two frame store buffers for later use in decoding P and B pictures. Decoding a P picture requires forming predictions from a previously decoded P or I picture. The decoded P picture is stored in a frame store buffer for use in decoding further P and B pictures. B pictures can require predictions from both of the frame store buffers. However, B pictures are not stored in the frame store buffers.

It will be appreciated that I and P pictures are not output from the temporal decoder as they are decoded. Instead, I and P pictures are written into one of the frame store buffers, and they are read out only when a subsequent I or P picture arrives for decoding. In other words, the temporal decoder relies on subsequent P or I pictures to flush previous pictures out of the two picture buffers. Accordingly, the spatial decoder of the present invention can provide a fake I or P picture when it is necessary to flush the temporal decoder's two frame store buffers. In turn, this fake picture is flushed when a subsequent video sequence begins.

As shown in Table 14, the picture frames are displayed in numerical order.

Display Order	I1	B6	B3	P4	B5	B6	P7	B8	B9	I10
Transmit Order	I	P4	B6	B3	P7	B5	B6	I10	B8	B9

Table 14: Frame Stores

However, in order to reduce the number of frames that must be stored in memory by the temporal decoder, the frames are transmitted in a different order. It is useful to begin the analysis from an intra frame (I frame). The I frame is transmitted in the order it is to be displayed. The next predicted frame (P frame), P4 is then transmitted. Then, any bi-directionally interpolated frames (B frames) to be displayed between the I frame and P4 frame are transmitted, represented by B6 and B3. This allows the transmitted B frames to reference a previous frame (forward prediction) or a future frame (backward prediction). After transmitting all the B frames to be displayed between I and P4, the P7 frame is transmitted. Next, all the B frames to be displayed between the P4 and P7 frames are transmitted, i.e., corresponding to B5 and B6. Then, the next I frame, I10, is transmitted. Finally, all the B frames to be displayed between the P7 and I10 frames are transmitted, corresponding to B8 and B9. This ordering of transmitted frames requires only 2 frames to be kept in memory by the temporal decoder at any one time, and does not require the decoder to wait for the transmission of the next P frame or I frame to display an interfecting B frame. As described above and shown in Table 14, the temporal decoder of the present invention can be configured to provide MPEG picture reordering. With this picture reordering, the output of P and I pictures is delayed until the next P or I picture in the data stream starts to be decoded by the temporal decoder.

As the P and I pictures are reordered, certain tokens, i.e. Picture\_Start, Picture\_Type, and Temporal\_Reference, are stored temporarily on the chip as the picture is written into the picture buffers. When the picture is read out for display, these stored tokens are retrieved. At the output of the temporal decoder, the DATA tokens of the newly decoded P or I picture are replaced with DATA tokens for the older P or I picture, and they are then sent to the video formatter. Note that the output from the temporal decoder is in tokenized macroblock format and there is no block-to-raster conversion.

In brief, the video formatter of the present invention stores two framestores or pictures. In some video formatters three pictures or framestores are used to accommodate such features as repeating or skipping pictures. The video formatter's off-chip DRAM holds three framestores. The use of three framestores here allows frames to be either repeated or skipped in situations where the frame rates of the decoded video and the display are different.

All I, B and P frames are stored in the framestores of the video formatter. At any one time, there may be one frame store from which data is being displayed, one frame store into which data is being written, and in video formatters with three framestores, one other frame may be being stored in the third frame store.

5 The present embodiment performs all the prediction, reordering and block-to-raster tasks MPEG normally handles by using a temporal decoder with two framestores and a video formatter with two framestores, i.e., for a total of four framestores. This is accomplished in the present invention by using a frame store sharing scheme that only uses three framestores. The present embodiment cannot, however, handle the repeat and skip frame tasks of a video formatter with only the  
10 three framestores.

The present invention stores I pictures in a first frame store and P pictures in a second frame store. Because of the need to perform the block-to-raster conversion, B frames are stored in the manner detailed below in a third frame store.  
15 In order to minimize the amount of external DRAM required, a scheme is used where successive B frames share the same third frame store.

When a B frame is decoded, it may refer to the two previously decoded I or P frames occupying the first and second framestores. The decoded B frame is written into the third frame store. The present embodiment allows the raster to  
20 commence prior to a frame store being completely filled. The raster is allowed to start before the frame store is filled so that the next B frame can be written into the same frame store to occupy the space vacated by the raster at the top of the previous frame.

In order to keep a record of which parts of the frame store are occupied with picture data, and which are available for new data, each frame store is split into sectors. In the present invention, each frame store is first split into two field stores, each of which comprises N sectors, where N is the number of block rows in the field.  
25

Frames coded as field pictures are straightforward. Each successive macroblock row occupies two sectors in a field store. Once the write back has progressed far enough down the frame, the raster starts reading out each sector from the top. Once the write back of the first frame has been completed, the start of the next frame is written into the space left by the raster. Checks on the status of each of the sectors ensures that the sector to be rastered is indeed full, and that for write back, the two sectors required are empty.  
30

Frames coded as frame pictures are more difficult. Unlike field pictures, the macroblock rows of data are not written to the DRAM in the same order as they are to be rastered. The field stores are written to in parallel, whereas the fields are rastered in turn.

5 Consider a picture with 8 sectors per field store. That is, Field store 0 consists of 8 sectors numbered 0 to 7, each of which contains one row of blocks (i.e., each 8 pixels deep by the width of the picture). Field store 1 consists of 8 sectors, numbered 8 to 15, each of which contains one row of blocks (i.e., each 8 pixels deep by the width of the picture).

10 The first macroblock row is written back into sector 0 in field store 0 and to sector 8 in field store 1. The field stores continue to be filled in parallel. At some point, the raster beings displaying sectors from field store 0, that point being chosen so that the raster of field store 0 does not catch up with the write back. However, the second frame cannot be written back in the same manner as the first. Because  
15 the sectors are written and read in a different order, waiting for the same two sectors to be free at the start of a frame would mean that write and read could not run continuously. This must be achieved in order to maintain the display and to maintain decoding at the necessary rate.

Accordingly, the second frame must be written into sectors of the frame store already freed by the raster. This is implemented by dividing the frame stores in two.  
20 Hence, for the second frame, the meanings of the half field stores change. Sectors 4-7 become the upper part of the second field store and sectors 8-11 become the lower part of the first field store, i.e., they swap over. The first macroblock row is written to sectors 0 and 4, once they are freed, with subsequent rows written to 1 and 5, then 2 and 6, and then 3 and 7. The next row is written to sectors 8 and 12,  
25 and so on through to 11 and 15. This reallocation to the memory is sufficient to allow the write back and raster to continue at the appropriate rate.

Should a third successive B frame arrive, the write back order reverts to that of the first frame.

30 In the shared B frame store, with FRAME pictures:

The FIRST picture is written back to -

Sectors 0 and 8 [1st macroblock row=2 block rows]

Then 1 and 9, 2 and 10, 3 and 11, ... 7 and 15.

The FIRST picture is rastered from -

35 Sector 0,

Then 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.

The SECOND frame is written to -

Sectors 0 and 4.

Then 1 and 5, 2 and 6, 3 and 7, 8 and 12, 9 and 13, 10 and 14, 11 and 15.

5 The SECOND frame is rastered from -

Sector 0.

Then 1, 2, 3, 8, 9, 10, 11, 4, 5, 6, 7, 12, 13, 14, 15.

10 Note that, in accordance with the present invention, the second frame, the first macroblock row is not written into sectors 0 and 1, which are, after all, the first two sectors to be freed by the raster. Instead, it waits for sector 4 to clear. This is done for two reasons: First, waiting for sector 4 to clear does not affect the system's ability to maintain continuous decoding and display, even in the situation of worst-case coded data, and it is easier to implement. Secondly, with picture sizes which divide into a number of sectors that are not a power of two, the sequence for writing to and reading from sectors of memory does not repeat often (for example, NTSC format has 30 sectors per field and the sequence would repeat every 58 frames).  
15 This makes testability and recovery difficult.

20 As far as implementation is concerned in the present invention, rather than keep a record of the status of each individual sector, each half field store is effectively implemented as a fifo, with pointers to the next location to be written and to be read. Thus, each fifo being full or empty causes write back and raster, respectively to be disabled. This makes use of the knowledge that each half field store is itself written and read only one way, just like a fifo.

25 In summary, the present invention, provides method for storing video information by providing video information in the form of an I Frame, a P Frame, a B<sub>1</sub> Frame and a B<sub>2</sub> Frame, storing the I Frame in a first Frame store, storing the P frame in a second frame store; providing a third Frame store having a first and second field store, the first and second field store being split into at least two memory areas respectively, storing the B<sub>1</sub> Frame in the third register, reading the  
30 B<sub>1</sub> Frame from a selected portion of the memory area in the first or second field store; writing a portion of the B<sub>2</sub> Frame into the selected portion of the memory area from which the B<sub>1</sub> Frame was read; whereby a reduced amount of memory can be used to store video information.

35 The two programs found herein below contain code to be used in the preferred embodiment of the invention. -

**Detailed Description of the Invention for a Parallel Huffman Decoder**  
In accordance with the present invention, the Parallel Huffman Decoder block will decode Huffman coded Variable Length Codes (VLCs) and Fixed Length Codes (FLCs), and pass through tokens under the control of the parser microprogrammable state machine (MSM).

This embodiment of the present invention handles both MPEG-2 as well as MPEG-1 Huffman codes. An important aspect of this embodiment of the invention is that it can sustain a high through-put due to the fact that it is a *parallel* decoder rather than a *serial* one.

This embodiment of the present invention uses a code lookup technique to decode Huffman codes. This is done to achieve the performance requirements and also to handle the second MPEG-2 transform coefficient table which is irregular or non-canonical in nature.

Furthermore, this embodiment of the invention has some features that allow it to decode certain more complex components from the stream in a single cycle without the assistance of an external controller. Examples of such complex components are Escape-coded coefficients, Intra-DC values and Motion Vector deltas, all of which are present in the stream as combined VLC/FLC components.

Referring now to Figure 52, there is shown how the Parallel Huffman Decoder 300 deals with variable length codes (VLCs). FLCs require a bypass mechanism which uses the selector 301 output to generate data and an input field to specify the length of the FLC. Thus, the ROM 302 is not required at all during FLC decoding.

However, to decode a VLC, input is first loaded into the two input data registers, 'MSReg' and 'LSReg' as shown in Figure 52. As the names imply, the "earlier" or most significant data is stored in MSReg. The selector is used to align the beginning of the next VLC with the ROM input. Thus, to decode the very first VLC, the selector outputs the top 28 bits of its 59-bit input and the top 16 bits of these are passed to the Huffman Code ROM 302. For subsequent VLCs, the selector effectively shifts the input according to the total count of bits decoded thus far. The count is maintained by adding the size of each VLC, as it is decoded, to a running total. The various word widths are a result of the maximum coded size which can be decoded, which is the 28-bit MPEG-1 Escape Coded Coefficient, and the maximum VLC size which is 16 bits (DCT coefficient tables).

The "table select" input is used to select between the various different Huffman code tables required by MPEG.



The Huffman Code ROM

The core of the implementation of the present invention, used to decode all the VLCs is a special ROM 302 whose addresses are controlled with a selector/shifter 301 as shown in Figures 52 and 53. The ROM 302 has the job of performing a VLC table index calculation, followed by the index-to-data operation that yields decoded data.

The index calculation can be thought of as a content addressable memory (CAM) operation with "don't care" matching implemented to handle the Huffman codes which form the presented data. Since all the VLC code tables are fixed, a CAM-ROM will suffice and this is the job of the ROM AND-plane shown in Figures 54 through 57. Since the index generation is performed in a look-up manner (rather than algorithmically) there is no restriction to handling tables which are canonical.

The ROM Or-plane converts the "index" (an activated word-line) into the decoded data and the size (or length) of the code. The data forms the decoded output (subject to error checking) and the size information is fed back to allow a calculation to be performed which controls the selector and, thus, presents the decoder ROM 302 with the correct data to perform the decoding of the next VLC in the subsequent cycle.

The ROM 302 address of the present invention is in two fields. The larger field is the bit-pattern to be decoded and the smaller field selects which Huffman code table is to be examined. The bit-pattern which must be examined is quite long, 18 bits, corresponding to the longest VLC code and there is an additional 4 bits of table select. Thus, there is a total address space of 20 bits (approximately one million addresses) although there are only in 450 entries in the ROM 302. The reason for the difference is due to the existence of "don't care" bits.

In order to decode VLCs, the AND-plane must be able to decode "don't care" bits in the VLC bit-pattern. This is because all VLCs which are shorter than the maximum 18 bits will be followed by additional bits which form no part of the decoding of that VLC. Because of the wide address, the AND-plane is predecoded (2->4), and the ROM 302 must combine "don't care" handling with this predecode. Furthermore, in addition to the complete MPEG code tables, the ROM 302 also has entries to identify illegal VLC patterns, which exist for some code tables.

Maximizing Throughput

In order to sustain output of one decoded item every cycle, some care must be taken to control the decoder input and special handling must be used for some "complex" symbols (i.e., ones which are not single FLCs or VLCs).

In order to sustain peak throughput of Escape-coded coefficients it must be possible to input at least one complete code per cycle. Since the maximum length required is 28 bits in MPEG-1 this dictates the input word width of 32 bits (being the next sensible size greater than 28).

Normal transform coefficients are also "complex" symbols, in the sense that they consist of a VLC followed by a 1-bit FLC which gives the sign of the level value and are handled in a similar manner to the other complex symbols (e.g. motion Vectors, Intra DC and Escape coded coefficients). Peak throughput cannot be achieved if coefficients are decoded as a VLC followed by an FLC (in separate cycles) and the alternative of allowing the ROM 302 to decode the sign bit would double the size of the two largest tables in the ROM. Thus in the present invention, special handling is used for various symbols so that a single cycle can produce the "final" required result.

#### FLCs and Tokens

The basis of FLC handling is to control the selector with the required length of the FLC and to bypass the ROM 302 and simply output the correctly selected FLC. Thus, simple FLCs are handled fairly naturally by the decoder, without significant extra hardware. Furthermore, tokens are not manipulated, but simply passed directly to the output of the decoder.

#### Implementation

This section describes several important features of the implementation of the decoder, in accordance with the present invention. The implementation includes the arrangement of registers with the counter 303 and selector 301, as shown in Figure 52, and the actual code ROM.

The schematic of Figure 53 shows how the core components are interconnected to implement the main Huffman decoding core section of the present invention. The registers  $ms[31:0]$  and  $ls[31:0]$  are MSReg and LSReg, respectively, and the block phselect is the selector. The counter logic is contained in the block phcclog (together with various other logic) and the count latch is called cntl[4:0]. The other logic on this schematic deals with handling commands, data and command dynamics, tokens, and the manipulation of the more "complex" symbols (performed in block phcop).

The schematic shown in Figure 54 illustrates a very small sample ROM design of the type used to implement the Huffman code ROM 302 in accordance with the present invention. The unusual features of this ROM 302 lie in the AND-plane where predecode and "don't care" handling are used to implement a method of decoding variable length Huffman codes.

Referring now to Figures 55, 56 and 57 and, more particularly to Figure 55, there is shown a first embodiment of a ROM AND-plane capable of "don't care" handling. In this embodiment, each address line (a[3], a[2], a[1] and a[0]) is driven across the AND-plane in both its true and inverted directions. To decode a "one" or a "zero" on a given address line, a transistor is connected to either the true or inverted address line in the conventional manner. In order to decode a "don't care" (denoted by x) a transistor is not connected to either the true or the inverted line.

Figures 56 and 57 show alternative embodiments that utilize pre-decoding to reduce worst-case number of series transistors in the decoding logic. In these examples, two address bits are combined together in predecoding such that one of four lines is driven high for each of the four possible numbers that can be represented with the two address bits. It will be appreciated by one of ordinary skill in the art that the present invention would work equally well with higher levels of predecoding in which more than two bits are combined together. If the two address bits that are grouped together in the predecoding have defined values (either 1 or zero, but not the "don't care") then a transistor is connected to the appropriate predecoded address line in the conventional manner. Similarly, if both of the address bits have a "don't care", then no transistor is used as before. However, if one of the address bits needs to have a defined value (1 or zero) whilst the other address bit requires "don't care", then the decoding requires that the wordline driven across the Or-plane be selected when either of two of the predecoded address lines is active. In the embodiment shown in Figure 56, this is achieved by placing two transistors, one on each of the relevant predecoded address lines, in parallel as shown in the case for the code; 001x. In the embodiment shown in Figure 57 the required decoding is achieved without using a parallel connection of transistors. In this case, two separate decodes are performed both of which must be selected. They are combined together using a NOR gate in the wordline driver such that the wordline is only activated if both of the selects are active.

The foregoing description is believed to adequately describe the overall concepts, system implementation and operation of the various aspects of the

invention in sufficient detail to enable one of ordinary skill in the art to make and practice the invention with all of its attendant features, objects and advantages. However, in order to facilitate a further, more detailed in depth understanding of the invention, and additional details in connection with even more specific, commercial  
5 implementation of various embodiments of the invention, the following further description and explanation is proffered.

Note that additional Figures, which are self explanatory to those of ordinary skill in the art, are included with this application for providing further insight into the detailed structure and operation of the environment in which the present invention  
10 is intended to function.

The aforedescribed pipeline system of the present invention satisfies a long existing need for further improvements in various aspects of video decoding systems, including an MPEG video decompression method and apparatus utilizing a plurality of stages interconnected by a two-wire interface arranged as a pipeline  
15 processing machine. Control tokens and DATA Tokens pass over the single two-wire interface for carrying both control and data in token format. A token decode circuit is positioned in certain of the stages for recognizing certain of the tokens as control tokens pertinent to that stage and for passing unrecognized control tokens along the pipeline. Reconfiguration processing circuits are positioned in selected  
20 stages and are responsive to a recognized control token for reconfiguring such stage to handle an identified DATA Token. A wide variety of unique supporting subsystem circuitry and processing techniques are disclosed for implementing the system, including memory addressing, transforming data using a common processing block, time synchronization, asynchronous swing buffering, storing of  
25 video information, a parallel Huffman decoder, and the like.

It will be apparent from the foregoing that, while particular forms of the invention have been illustrated and described, various modifications can be made without departing from the spirit and scope of the invention. Accordingly, it is not intended that the invention be limited, except as by the appended claims.

WE CLAIM:

1. An apparatus for synchronizing time, comprising:
  - a time stamp for determining presentation time;
  - a clock reference for initializing system time in a first circuit;
  - 5 a first time counter in communication with said clock reference for keeping system time in a first circuit; and
  - a second time counter initialized by said clock reference in a second circuit synchronized with said first time counter, for keeping a local copy of said system time and for determining the presentation timing error between said local copy of system time and said system time by comparing said time stamp to said second time counter.
2. An apparatus for synchronizing a system decoder and a video decoder, comprising:
  - a system decoder;
  - a time stamp for determining display time;
  - 5 a clock reference for initializing system time in said system decoder;
  - a first time counter in communication with said clock reference for keeping system time in said system decoder; and
  - a second time counter initialized by said clock reference in said video decoder synchronized with said first time counter, for keeping a local copy of system time and for determining the display timing error between said local copy of system time and said system time by comparing the time stamp to said second time counter.
3. An apparatus for synchronizing a first circuit and a second circuit, comprising:
  - a clock reference for initializing system time in a first circuit, said first circuit having a time counter in communication with said clock reference for keeping
  - 5 system time;
  - a first elementary stream time counter in said first circuit for providing elementary stream time;
  - said first circuit being adapted to receive a time stamp, and said first circuit adapted to generate synchronization time by adding elementary stream time to said
  - 10 time stamp and subtracting system time; and
  - said second circuit being adapted to receive synchronization time from said first circuit and having a second elementary stream time counter in synchronization

with said first elementary stream time counter for providing a local copy of said elementary stream time and for determining a timing error between said system time  
 15 and said time stamp by comparing synchronization time to said local copy of said elementary stream time;

whereby said clock reference signal does not have to be passed directly to said second circuit in order to determine timing error.

4. An apparatus for synchronizing a first circuit and a second circuit, comprising:

a clock reference for initializing system time in a first circuit;

said first circuit having a time counter in communication with said clock  
 5 reference for keeping system time;

a first video time counter for providing video decoding time;

said first circuit being adapted to receive a video time stamp and generate synchronization time by adding video decoding time and video time stamp and subtracting system time; and

10 said second circuit being adapted to receive synchronization time from said first circuit and having a second video time counter in synchronization with said first video time counter for providing a local copy of video decoding time and for determining a timing error between said system time and said video time stamp by comparing said synchronization time to said local copy of video decoding time;

15 whereby said clock reference signal does not have to be passed directly to said second circuit in order to determine timing error.

5. A method for providing timing information, comprising the steps of:

providing a video data stream having a time stamp carried in packet header said time stamp referring to the first picture in a packet of data;

providing a register having a flag for indicating valid time stamp information

5 which is taken from said packet header and placed into the register;

removing said time stamp from said video data stream and placing it in said register; and

encountering a picture start and subsequently examining the status of said register to determine if valid time stamp information is contained in said register by checking flag status.

6. A process for decoding video comprising the steps of:

determining display time errors against a threshold value;

parsing video data into tokens for further processing;

- determining if a time stamp token is indicated;  
 5 comparing the time stamp token to a video time;  
 generating a compared value to determine an indication of timing error;  
 determining whether the compared value, when compared against a  
 threshold value, is within acceptable parameters when a timing error is indicated;  
 and  
 indicating when the compared value is outside acceptable parameters.
7. An apparatus for using a system decoder and a video decoder,  
 comprising:  
 a system decoder adapted to accept MPEG system streams and  
 demultiplexing video data and a video time stamp from a stream;  
 5 said system decoder having a first time counter representative of system  
 time;  
 a video decoder for accepting said video data and said video time stamp;  
 said video system having a second time counter in synchronization with said  
 first time counter; and  
 10 said video decoder also having a video decoder buffer for accepting said  
 video data at a substantially constant rate and outputting said video data at a  
 varying rate and for passing a video time stamp.
8. A method for determining a timing error between a first circuit and a  
 second circuit, comprising the steps of:  
 providing the first circuit with a system time (SY), a time stamp (TS), and an  
 elementary stream time (ET);  
 5 obtaining synchronization time (X) by using the elementary stream time (ET),  
 the time stamp (TS) and the system time (SY), in accordance with the equation,  
 $X = ET + TS - SY$ ;  
 providing synchronization time (X) to the second circuit; and  
 generating a synchronized elementary stream time (ET2);  
 10 obtaining a timing error by using synchronization time (X) and in accordance  
 with the equation  $ET2 - X$ ;  
 whereby the first circuit can be synchronized with the second circuit without  
 passing system time to the second circuit.
9. A method for determining a timing error between a first circuit and a  
 second circuit, comprising the steps of:  
 providing the first circuit with a time stamp (TS), and an initial time (IT);

- obtaining synchronization time (X) by using the time stamp (TS) and the initial  
 5 time (IT), in accordance with the equation  $X=TS-I$ ;  
 providing synchronization time (X) to the second circuit;  
 generating a synchronized elementary stream time (ET); and  
 obtaining a timing error by using synchronization time (X) and in accordance  
 with the equation  $ET-X$ ;
- 10 whereby the first circuit can be time synchronized with the second circuit  
 without passing time to the second circuit.
10. A method for determining a timing error between a first circuit and a  
 second circuit, comprising the steps of:  
 providing the first circuit with a system time (SY), a video time stamp (VTS),  
 and a video decoding time (VT);
- 5 obtaining synchronization time (X) by using the video decoding time (VT), the  
 video time stamp (VTS) and the system time (SY), in accordance with the equation,  
 $X=VT+VTS-SY$ ;  
 providing synchronization time (X) to the second circuit;  
 generating a video decoding time (VT2) in the second circuit which is  
 synchronized to the video decoding time (VT) in the first circuit; and  
 obtaining a timing error by using synchronized time (X) and in accordance  
 with the equation  $VT2-X$ ;
- whereby, the first circuit can be time synchronized with the second circuit  
 without passing system time to the second circuit.
11. A method for addressing memory, comprising the steps of:  
 providing a fixed width word having a predetermined fixed number of bits to  
 be used for addressing variable width data;
- 5 defining the fixed width word with a width defining field and an address field;  
 providing the width defining field with at least one bit to serve as the  
 termination marker;
- defining the address field with a plurality of bits defining the address of the  
 data;
- 10 varying the size of bits in the address field in inverse relation to the size of the  
 variable width data;
- varying the number of bits in the width defining field in direct relation to the  
 size of the variable width data; and



maintaining a fixed width word for addressing variable width data while varying the width of the width defining field and the address field.

12. A method for addressing memory, comprising the steps of:

providing a fixed width word having a predetermined fixed number of bits to be used for addressing data;

defining the fixed width word with an address field and a substitution field;

5 defining the address field with a plurality of bits defining the address of the data;

defining a variable width substitution field with at least one substitution bit; the substitution field having at least one bit to serve as a termination marker

between the address field and the substitution field;

10 using the substitution field to indicate substituted bits from a separate addressing source; and

maintaining a fixed width word for addressing variable width data while inversely varying the width of the address field and the width of the substitution field.

13. A method for addressing variable width data in a memory, comprising the steps of:

providing memory having words of predetermined width and composed of partial words;

5 rotating the partial word to be accessed to at least significant bit justification; extending remaining part of the word so that the accessed word will be recognized as the partial word;

restoring the remaining part of the word; and

rotating the word until the partial word is restored to its original position.

14. A parallel Huffman decoder, comprising:

a selector;

a pair of input registers for receiving Huffman coded data, both of said registers directing input in parallel to said selector; and

5 a Huffman Code ROM for receiving input from said selector and another ROM table select input; said ROM providing decoded data output.

15. A RAM interface for connecting a bus to RAM, comprising:

means for receiving from a bus a plurality of data words, and buffering the received data words;

5 means for receiving from said bus an address associated with said plurality of data words;

means for generating a series of addresses in RAM into which the buffered data words will be written, the series of addresses being derived from the received address; and

means for writing said buffered data words into RAM at the generated addresses.

16. A RAM interface for connecting a bus to RAM, comprising:

a plurality of data words stored in RAM at predetermined addresses;

means for receiving from a bus a RAM address associated with said plurality of data words;

5 means for generating a series of RAM addresses for addressing said plurality of data words in said RAM, the series of addresses being derived from the received address;

means for buffering data words read from said RAM; and

10 means for reading from said RAM said plurality of data words, using said series of RAM addresses generated by said address generating means, and writing said data words into said buffer means.

17. A method for controlling the buffering of encoded video data organized as frames, comprising:

determining the picture number of a frame;

determining the desired presentation number of said frame; and

5 marking a buffer as ready when said picture number is on or after said desired presentation number.

18. An apparatus for transforming data, comprising:

A first latch defining a first data stream source and a second latch defining a second data stream source;

5 said first and said second latches being in communication with an arithmetic unit;

said arithmetic unit communicating data to a transposer;

said transposer transposing and communicating said data to said second latch;

said second latch being arranged to absorb data; and

10 said second latch and said first latch communicating said first and second data streams in an interleaved manner to said arithmetic unit, further defined that in said communication in the interleaved manner said second latch does not interrupt communication from said first latch;

111

whereby a common arithmetic unit is used for said first and said second data streams.

19. A process for transforming data using a common arithmetic unit, comprising the steps of:

loading the data in a first latch and upon reaching a predefined number of cycles transmitting the data to the arithmetic unit and loading a first maker bit into  
5 a control shift register;

loading data into a second latch, the second latch being adapted to absorb data;

transmitting the data in the second latch to the arithmetic unit when the first control shift register reaches a predetermined state and the second latch is filled  
10 with a predetermined amount of data;

failing to transmit data from the second latch, if the second latch is not filled with a predetermined amount of data; and

recovering the second latch when the first latch is receiving non-data.

20. A swing buffer apparatus, comprising:

at least two memory arrays;

a write control circuit in communication with said memory arrays for controlling data input into said memory arrays;

5 a read control circuit in communication with said memory arrays for controlling data output from said memory arrays; and

said write control circuit and said read control circuit being in communication to allow a synchronized control of said memory arrays.

21. A method of asynchronously accessing cells in a memory, comprising the steps of:

using a decoder to decode at least a pair of cells; and  
reading one of the cells and writing to the other of the cells.

22. A method for storing video information, comprising the steps of:

providing video information in the form of an I Frame, a P Frame, a B<sub>1</sub> Frame and a B<sub>2</sub> Frame;

storing the I Frame in the first Framestore; storing the P frame in a second  
5 framestore; providing a third Framestore having a first and a second field store, the first and second field store being split into at least two memory areas respectively; and

storing the B<sub>1</sub> Frame from a selected portion of the memory area in the first or second field store; writing a portion of the B<sub>2</sub> Frame into the selected portion of the memory area from which the B<sub>1</sub> Frame was read;

whereby a reduced amount of memory can be used to store video information.

23. A memory for "don't care" handling, comprising:

a set of memory address lines, inverted address lines and data lines, said address lines and inverted address lines being connected according to a decoding format to access the addressed information in the form of a data word, "don't care" address locations on the data lines being unconnected with the associated address lines and said inverted address lines.

24. A method of accessing Dynamic Random Access Memory (DRAM) to store and retrieve data words associated with a two dimensional image, the DRAM including two separate banks, each bank being capable of operating in page mode to read and write the data words, the two dimensional image being organized in two dimensional grid pattern of cells, each cell containing an M by N matrix of pixels, and the words associated with each cell occupying one page or less of a bank, the method comprising the steps of:

(a) assigning each cell a particular one of the two banks so that all data words associated with that particular cell are read from and written to one particular page of that particular bank, the assignment of banks to cells being done such that each cell is associated with a different bank than any bordering cell which is also either in the same row or in the same column; and

(b) reading the data words associated with a cell that is composed of a matrix of pixels, and that is not aligned with the two dimensional grid pattern, but that is aligned with pixels in cells in the two dimensional grid pattern.

25. A method for accessing Dynamic Random Access Memory (DRAM) to store and retrieve data words associated with a two dimensional image, the DRAM including two separate banks, each bank being capable of operating in page mode to read and write the data words, the two dimensional grid pattern of cells, each cell containing an M by N matrix of pixels, and the words associated with each cell occupying one page or less of a bank, the method comprising the steps of:

(a) assigning each cell a particular one of the two banks so that all data words associated with that particular cell are read from and written to one particular page of that particular bank, the assignment of banks to cells being done

- 10 such that each cell is associated with a different bank than any bordering cell which is also either in the same row or in the same column; and

(b) reading the data words associated with a cell that is composed of an M by N matrix of pixels, and that is not aligned with the two dimensional grid pattern, but that is aligned with pixels in cells in the two dimensional grid pattern.

26. The method of claim 63, wherein the DRAM includes a first and second bank, and the step (b) of reading the data words associated with the unaligned cell includes the steps of:

- (c) reading, from the first bank of DRAM, the data words  
5 associated with one of the cells in the grid pattern containing data words associated with the unaligned cell;

(d) reading, from the second bank of DRAM, the data words associated with another of the cells in the grid pattern containing data words associated with the unaligned cell;

- 10 (e) repeating steps (c) and (d) until all the data words associated with the unaligned cell have been read.

27. A method of accessing from RAM a number M of words that is less than the predetermined fixed burst length N of the RAM, the RAM including an enable line that selectively enables and disables reading from and writing to the RAM, the method comprising the steps of:

- 5 ordering N words to be read from or written to the RAM;  
determining when M words have been read from or written to the RAM, M being less than N; and  
disabling the RAM upon determining M words had been read from or written to the RAM.

28. A method of reading from RAM a number M of words that is less than the predetermined fixed burst length N of the RAM, the RAM including an enable line that selectively enables and disables reading from the RAM, the method comprising the steps of:

- 5 ordering N words to be read from the RAM;  
determining when M words have been read from the RAM, M being less than N; and

disabling the RAM upon determining M words had been read from the RAM.

29. A method of writing to RAM a number M of words that is less than the predetermined fixed burst length N of the RAM, the RAM including an enable line

that selectably enables and disables writing to the RAM, the method comprising the steps of:

- 5     ordering N words to be written to the RAM;
- determining when M words have been written to the RAM, M being less than N; and
- disabling the RAM upon determining M words had been written to the RAM.

The following more detailed description of the system of the present invention is set forth for purposes of organization, clarity and convenience of explanation under the headings listed below:

Overview .....	
Start Code Detector .....	
Parser .....	
Spatial Processing .....	
Predictions .....	
Display Circuitry .....	
Parallel Start Code Detector (sodp) .....	
Input FIFO .....	
Input Circuit .....	
Start Codes .....	
Removal of bit stuffing .....	
Search modes .....	
Non-aligned start codes .....	
Overlapping Start Codes .....	
Unrecognized Start Codes .....	
Extension and User Data .....	
insertion of PICTURE_END Tokens .....	
Stop After Picture Interrupt .....	
discard_all .....	
Access Bit .....	
Tokens Recognized by sodp .....	
Sodp Memory Map .....	
Implementation .....	
DataFlow Around the Coded Date Buffer .....	
Theory of Operation .....	
Discontinuities .....	
Start-up .....	
Embodiment .....	
Hardware .....	
MSM handling of Time-Stamp Information .....	
Start-Up .....	
MSM Time-stamp error codes .....	
Support for 30 Hz .....	
Introduction .....	
State Machine .....	
Jumps and Calls .....	
Interrupts and errors .....	
Jump addresses .....	
State Machine Internal Instructions .....	

State Machine testing .....	
State Machine ucode map .....	
State Machine ucode word .....	
Arithmetic Core .....	
ALU .....	
Shift block .....	
Carry block .....	
Condition block .....	
ALU core .....	
ALU ucode word .....	
Use of the ALU .....	
Register File .....	
Register file addressing .....	
Register file register types .....	
Register file address map .....	
Register file ucode word .....	
Token Port .....	
Token Port ucode word .....	
Multiplexers .....	
UPI Memory Map .....	
Introduction .....	
Interfaces .....	
Functional Description .....	
Timing requirements .....	
Microprocessor Interface Access .....	
Introduction .....	
Interfaces .....	
Functional description .....	
Mal-formed tokens .....	
Zig-zag scan paths .....	
Raster scan order .....	
Microprocessor Interface Access .....	
Introduction .....	
Prediction in frame pictures .....	
Frame-based prediction .....	
Field-based prediction (in a frame picture) .....	
Dual prime (in frame pictures) .....	
Prediction in field pictures .....	
Field-based prediction .....	
16x8 MC .....	
Dual prime in field pictures .....	
Overall organization .....	
Horizontal Upsampler .....	



Introduction .....	
4:3 Upsampling .....	
3:2 Upsampling .....	
2:1 Upsampling .....	
Boundary Effects .....	
The number of output pels .....	
Position signals .....	
Multiplexed data .....	
Horizontal Alignment .....	
Upsampling ratio .....	
Video Timing Generator .....	
Introduction .....	
Horizontal Timing .....	
Vertical Timing - PAL .....	
Vertical Timing - NTSC .....	
VTG Structure .....	
Horizontal Machine .....	
Vertical Machine .....	
Hardwired Comparator Design .....	
Output multiplex .....	
Border generation .....	
Vertical Border .....	
UPI controls .....	
Output multiplex .....	

### Overview

This detailed description deals with the present invention as an entire chip. Referring now to Figure 58, there is shown a very high level block diagram of the system. In subsequent sections, each block is expanded to provide a more detailed block diagram.

This description accurately documents all of the interfaces between the various functional blocks of circuitry. This should allow each block to be designed with a complete knowledge of the interfaces that it is expected to provide.

As shown in Figure 58, the primary system components include clock generator 350, a start code detector 201, a parser 202, a microprocessor interface 320, a memory control subsystem 352, a spatial processing subsystem 351, a predictions subsystem 208 and a display 355. Figure 58 further illustrates the interfacing that occurs between the various system components.

### Start Code Detector

Figure 59 shows the start code detector 201 (SCD) interfaces with other blocks of circuitry of the system in accordance with the present invention.

The SCD 201 can be thought of as providing three distinct functions. First, the SCD 201 provides an input circuit that receives data either from dedicated pins or from the MPI 320. Second, the SCD 201 detects start codes in the data, and third, the SCD provides the necessary circuitry to assemble the incoming data into a format to be used internally within the coded data buffer (CDB) 321.

### Parser

Figure 60 illustrates the parser subsystem, in accordance with the present invention. Data which was formatted for the CDB 321 is unpacked and passed to the parser which receives instructions from the MPI 320. Thereafter, the data is passed via a two-wire interface to the rest of the system.

### Spatial Processing

Figure 61 illustrates the components of the spatial processing circuitry. These components include an inverse modeler (Imodel) 325, an inverse zig-zag (IZZ) 326 and inverse quantizer (Iquant) 327 and an inverse discrete cosine transfer (IDCT) 328. The data passes into the Imodel 325, then to the IZZ 326, next to the Iquant 327 and then to the IDCT 328.

### Display Circuitry

The display circuitry of the present invention is shown in Figure 62. This system includes a vertical upsampler 210, a horizontal scale subsystem 331, an output multiplexer 332 and a video timing generator 333.

### Parallel Start Code Detector (scdp)

The start code detector 201, in accordance with the present invention, is a parallel start code detector, i.e., it passes data in parallel. This system is similar to that previously disclosed in British Application Serial No. 9405914.4 filed March 24, 1994, and EP0 Application Serial no. 92306038.6 filed June 30, 1992, (hereinafter "Brolly"). However, several major differences exist between the two start code detectors. First, byte alignment is assumed. There is no shifting of the data in order to find start codes in the present invention. Second, the present invention operates primarily with MPEG data.

An MPEG (1 and 2) start code consists of a unique bit (byte pattern) in the bit stream known as the start\_code\_prefix. The pattern is 23 zeros followed by a one. The 8 bits immediately following the start\_code\_prefix is known as the start\_code\_value. This indicates the type of the start code. Start codes arriving at the SCD of the present invention are required to be byte aligned. Accordingly, the above data can be specified as a byte sequence. For example:

```
0x00
0x00
0x01
0xb8
```

is a group\_start code.

### Input Fifo

The present invention is designed such that given a peak data rate of 250 Kbytes/s and assuming that the coded data buffer does not overflow, the in\_accept pin will never be pulled low. Hence, to calculate the length of the input fifo, it is necessary to know: 1) the worst case wait time for a swing buffer, and 2) the worst case data expansion through the SCD.

With the input data arriving at the coded data clock rate, in accordance with the present invention, scdp will generate two stalls per start code (having removed three bytes from the data stream).

### Input Circuit

The input circuit of the present invention performs exactly the same way as is disclosed in Brolly. However, there are a few differences of note between the two circuits. First, the up won't be made to wait until a valid end of a token (cause this may never set). Instead,

it will be made to wait until a signal in\_token is low. Second, generation of the DATA header, when entering byte mode, depends on there being some byte mode data.

### Start Codes

In the present invention, the MPEG start codes are recognized and converted to tokens by the SCD. These are shown in Table 15.

Start Code Type	Start Code Value
picture_start_code	0x00
slice_start_code	0x01 to 0xaf
reserved	0xb0
reserved	0xb1
user_data_start_code	0xb2
sequence_start_code	0xb3
sequence_error_code	0xb4
extension_start_code	0xb5
reserved	0xb6
sequence_end_code	0xb7
group_start_code	0xb8

Table 15. Start Code Values

### Removal of Bit Stuffing

Any zero bits preceding a start\_code\_prefix are stuffing and can safely be removed. In the present invention, only complete bytes of stuffing are removed.

For example, in the byte sequence shown below there are 13 stuffing bits, only 8 of which are actually removed.

0x20 // 5 stuffing bits  
0x00 // 8 stuffing bits

0x00  
 0x00  
 0x01 // start\_code\_prefix

### Search Modes

The search\_modes, in accordance with the present invention, are described as follows in Table 16:

Search_mode	Operation
0	Normal Operation
1	Search for picture_start or higher
2	Search for group_start or higher
3	Search for sequence_start or higher

Table 16. Search Modes

Any non-zero search mode causes all arriving data to be discarded until the desired class of start code is found. At this point, the search mode is reset to zero, and a start\_code\_search interrupt may be generated. A new control bit, stop\_on\_search, determines whether the SCD actually stops after generating the interrupt (the interrupt is also masked in the usual way, but stopping is not mandatory).

In the present invention, search\_mode is also set to zero if the SCD receives a FLUSH token. However, when the FLUSH token terminates discard\_all, search\_mode is completely reset, i.e., search\_mode is reset by the combination of a FLUSH token and discard\_all.

### Non-aligned Start Codes

Any run of more than one zero bytes followed by a 0x01 is a start code. Furthermore, any run of greater than 23 zeros NOT followed by a one is a non-

aligned start code. In the byte aligned world, this translates to: If, after removal of bit stuffing, 0x0f is not received, then the start code was non-aligned. Note that this statement actually misses some non-aligned start codes (where less than a byte of stuffing was involved).

Rather than going to the effort of describing in the data sheet which classes of non-aligned start codes are detected, the scdp of the present invention ignores them. In other words, stuffing is still removed.

### Overlapping Start Codes

It is possible for the "value" part of a start code to form part of the "prefix" of a subsequent start code. This typically occurs for two reasons: 1) the standard allows system level start codes to occur anywhere in the stream - including directly in the middle of a video level start code, and 2) errors. Removing all erroneous looking start codes until the last one provides a better chance of error recovery.

In the byte aligned environment, in accordance with the present invention, the only way an overlapping start can happen is if a picture\_start (value = 0x00) forms part of another start code. In this scenario, the picture\_start is removed from the data, and the second start code is decoded. If this, in turn, is overlapped then the same procedure applies until a non-overlapping start code is detected.

### Unrecognized Start Codes

In the present invention, the reserved values (0xb0, 0xb1, 0xb6), all system start codes (0xb2 to 0xb7), and the sequence\_error code (0xb4) are each treated as unrecognized start codes. After removing the unrecognized start code, the SCD discards all incoming data until the next valid start code is found. It will also set the unrecognized\_start error register and, depending on the unrecognized\_start mask, will generate an interrupt.



### Extension and User Data

Two configuration bits are used in the present invention.

- 1) Discard\_user (or not)
- 2) Discard\_extn (beyond MPEG2 main profile, main level)

Both of these configuration bits are reset to ONE.

MPEG2 extn start codes are different. The four bits following the extension\_start\_value are now an extension\_start\_code\_identifier and must be decoded by the SCD. Four new tokens are generated to flag these. The allowed extension\_start\_code\_identifiers and their respective tokens are shown in Table 17. However, reserved extension\_start\_code identifiers are not recognized. Unrecognized extension\_start\_codes are either discarded (depending on Discard\_extn), or replaced with the (old) extension\_data token.

extension_start_code_identifier	Name	New Token	Head
0000	reserved		
0001	Sequence Extension ID	SEQUENCE_EXTN	0xab
0010	Sequence Display Extension ID	SEQUENCE_DISPLAY_EXTN	0xab
0011	Quant Matrix Extension ID	QUANT_MATRIX_EXTN	0xab
0100	reserved		
0010	Sequence Scalable Extension ID		
0110	reserved		
0111	Picture Pan Scan Extension ID		
1000	Picture Coding Extension ID	PICTURE_CODING_EXTN	0xab
1001	Picture Spatial Scalable Extension ID		
1010	Picture Temporal Scalable Extension ID		

1011 to 1111	reserved		
--------------	----------	--	--

Table 17. MPEG2 extension\_start\_code\_identifiers

### Insertion of PICTURE\_END Tokens

None of the current standards (MPEG1,2, JPEG, or H.261) specify a way of ending a current picture.

However, in the present invention, the SCD 201 maintains a piece of state called `in_picture`. This state is set whenever a `PICTURE_START` token is output by the SCD 201. Any subsequent start code that is higher in the syntax than `picture_start` (or a `FLUSH` token) causes the generation of a `PICTURE_END` token. The `PICTURE_END` token is generated and output before any tokens associated with the new start code. `in_picture` is reset when the `PICTURE_END` token leaves the SCD 201. If the SCD 201 receives tokens in the input data stream, the action is logically identical - including receiving a `PICTURE_END` token. In summary, the start codes (and tokens) which may cause a `PICTURE_END` to be generated, in accordance with the present invention, are:

`picture_start_code` OR token  
`group_start_code` OR token  
`sequence_start_code` OR token  
`sequence_end_code` OR token  
`FLUSH` token

### Stop After Picture Interrupt

The stop after picture (sap) feature is of the present invention functions to facilitate a clean way of finishing off a current sequence, e.g., channel change. It is necessary to accomplish this function as automatically as possible and without the need for external real time software.

The sap control bit is referred to as a `flag_picture_end`.

There are two control bits in addition to the flag\_picture\_end, mask, and error bits:

- 1) **after\_picture\_stop**: Determines whether, after generating the interrupt, the SCD stops.
- 2) **after\_picture\_discard**: Having generated a flag\_picture\_end interrupt, this bit determines whether scdp automatically goes into discard\_all mode.

In this way, discard\_all mode doesn't need to know what event called it and it is possible to leave the discard\_all mode and to proceed to a search mode quickly and cleanly.

In accordance with the invention, whenever a PICTURE\_END token is output by the SCD, the flag\_picture\_end bit determines whether any action is taken. If flag\_picture\_end is set, a FLUSH is generated after the PICTURE\_END, and the event is generated. Interrupting depends on the flag\_picture\_end\_mask, and (having interrupted) stopping depends on after\_picture\_stop.

By way of example, for a channel change, the sequence of events is as follows:

- 1) Set flag\_picture\_end with after\_picture\_stop = 0 and after\_picture\_discard = 1.
- 2) Respond to flag\_picture\_end\_event.
  - a) Set search mode to sequence (for example).
  - b) Retune, etc.
- 3) Either FLUSH or s/w resets discard\_all.
- 4) scdp searches for the start of the next sequence.

#### discard\_all

An R/W control bit, discard\_all, causes the scdp of the invention to discard all input up to and including a FLUSH token. This bit is automatically reset by a FLUSH token and may be set by the flag\_picture\_end function.

### Tokens Recognized by scdp

While the primary function of most of the scdp of the present invention is related to actual token generation, there are, several tokens which when applied to the coded data port (or via the input circuit) are decoded and acted upon by the scdp. Table 18 illustrates and defines these tokens.

Token	Header	Action	Comments
FLUSH	0x17	Flushes scdp	These tokens may cause the generation of a PICTURE_END. In this case, they would reset In_picture and may cause a flag_picture_end event and a FLUSH to be generated.
PICTURE_START	0x12	Sets in_picture	
PICTURE_END	0x16	Resets in_picture	
GROUP_START	0x11		
SEQUENCE_START	0x10		
SEQUENCE_END	0x14		
DATA	0x04 etc.	Data is searched for start codes	
Other	-	Unrecognized tokens are passed through scdp unchanged	

Table 18. Recognized Input Tokens

### Scdp Memory Map

The various registers and their associated addresses for the scdp of the present invention are described in Table 19.

Register Name	Bits	Reset	Comments	Address
scdp_access		0		0x0
scdp_access	[0]	0	Access bit	
scdlpc_cdo[7:0]				0x1

Register Name	Bits	Reset	Comments	Address
CD0[7:0]	[7:0]		upi coded data port	
sddpc_cd1[7:0]				0x2
coded_busy	[7]	1	Read Only	
enable_coded	[6]	0		
coded_extn	[7]		Read Only	
sddp_ctl0[7:0]		0x30		0x03
discard_extn	[5]	1		
discard_usr	[4]	1		
discard_all	[3]	0	Reset by FLUSH	
flag_picture_end	[2]	0	Enables event	
after_picture_stop	[1]	0	Only if event enabled	
after_picture_discard	[0]	0	Only if event enabled	
sddp_ctl1[7:0]		0		0x4
stop_after_search	[2]	0	Only if event enabled	
start_code_search[2:0]	[1:0]	0		
sddp_event[7:0]		0		0x5
end_search_event	[0]	0		
unrecognized_start_error	[1]	0		
flag_end_lof_picture_event	[0]	0		
sddp_mask[7:0]		0		0x6
end_search_mask	[2]	0		
unrecognized_start_mask	[1]	0		

Register Name	Bits	Reset	Comments	Address
flag_end_of_picture_mask	[0]	0		

Table 19. Parallel Start Code Detector Memory Map

### DataFlow Around the Coded Data Buffer

The present invention provides the following advantages:

- 1) A method of forcing the buffer to swing.
- 2) A way of avoiding having to pack bytes into an odd number of bits.
- 3) Reducing the width of the (potentially long) bus of the SCD down to 8 bits.
- 4) The SCD does its own packaging into 32 bit data. To avoid a large bus, this 32-bit of the SCD sits inside the dramif. In the present invention, it is referred to as sccobin. This module packs all DATA into 32 bit words, dead-reckoning in between non-DATA tokens.
- 5) The swing buffers do their own counting and swinging. The buffers flush in response to a signal, fill\_and\_swing, from sccobin in response to a PICTURE\_END or a FLUSH token (or signal).
- 6) The unpacking module, sccobout, which sits prior to the Huffman Decoder, deletes all data following a FLUSH or PICTURE\_END until it receives a buffer\_start signal provided by the output swing buffer.

## Introduction

This section defines the handling of time-stamp information, in accordance with the present invention.

## Theory of Operation

In MPEG-2 video and audio, data is synchronized using information carried in the MPEG-2 systems stream. There are essentially two types of information that deal with synchronization; clock references and time-stamps.

Clock references are used to inform the decoder what number is used to represent the time "now". This is used to initialize a counter that is incremented at regular intervals so that the decoder has, at all times, a notion of what the current time is.

Time-stamps are carried for each of the streams of data that are used to make up the program (typically video and audio). In the case of video, a time-stamp is associated with a picture and it tells the decoder at what "time" (defined by the counter that was initialized by the clock reference) it should display the picture.

However, as with all things in MPEG, the situation is rather more complicated than this. There are two types of clock references; Program Clock References (PCRs) and System Clock References (SCRs). Clock has information to a resolution of 90 kHz while the other clock has additional information to extend the resolution to 27 MHz. Clock references are included in the data stream fairly often in order that "time" may be reinitialized after a random access or channel change.

There are also two types of time-stamps: Presentation Time-Stamps (PTSs) and Decoder Time Stamps (DTSs). These only differ for I-pictures and P-pictures which have to be reordered (not B-pictures). The DTS tells you when to decode the picture, whereas the PTS tells you when to display it. In the simple case of frame pictures with no 2-3 pull-down effects, the difference between DTS and PTS of an I-picture or P-picture will be one more than the number of B-pictures that follow that picture frame periods.

The important complication to appreciate is that the DTS and PTS refer to a hypothetical model of a decoder that can decode pictures instantly. Any real decoder cannot do this and must take steps to modify the theoretical time that it should display pictures (defined by the time-stamps and the clock references). This modification will depend on the details

of the architecture of the decoder. Clearly any delay introduced by the video decoder must be matched by an equivalent delay in the audio decoder.

#### Discontinuities

Discontinuities in the concept "time" may occur. For instance, in an edited bitstream each edit point will have discontinuous time. A similar situation occurs at a channel change. Care must be taken because using a time-stamp that was encoded in one time regime with respect to a "time" defined by a clock reference from another regime will clearly lead to incorrect results.

#### Start-up

A particular problem occurs at start-up (or channel change) because there are two potentially competing requirements for starting to decode correctly. For video considerations, it is now necessary to start decoding with an I-picture that follows a system header (this may not be true in all situations, but is largely a correct statement) but for system considerations the first decoded picture ought to carry a time-stamp. However, there is no requirement that every picture carry a time stamp and, therefore, it is possible that one may wait for ever if they try and look for a picture that is both an I-picture and carries a time-stamp.

One might think of calculating what the time-stamp would have been for an I-picture from a picture that precedes it that does have a time-stamp. Unfortunately this is very difficult to do because it would be necessary to partially decode the intervening pictures to determine whether they are field or frame pictures (and whether `repeat_first_field` is set). This requires that the data go through the coded data buffer and be discarded by the Huffman Decoder.

#### Embodiment

Figure 63 shows a first embodiment for implementing time-stamp management. The clock references 253 are decoded by the system demultiplex 254 of the present invention and placed into a counter 255, incremented at 90 kHz, that represents time. They are also loaded into a second copy of the counter 258 that is located inside the video decoder 270.



The time-stamps flow through the video buffer 271 so that they are delayed by the same amount as the video data. These are then compared with the local copy of time to determine whether the picture is too early or too late.

Another embodiment, in accordance with the present invention, is shown in Figure 64. This avoids the need for the clock references 253 to be passed to the video decoder 270. This is achieved by using a second counter "vid\_time" 272, 273 which is maintained both in the video decoder 270 and the system decoder 256. They are reset at power on and then free run from there on. Since this embodiment requires that the two counters stay in step, it is necessary to take steps to ensure they do not get out of step. This can be accomplished using carry out of the counter in the system demux to reset the one in the video decoder (as shown).

Another advantage of this embodiment is that there is no need for the full 33 bits of the number to be dealt with. The ideal would be to restrict the counters to 16 bits to allow 16 bit handling on the video decoder 270. Although this would appear to represent an insufficient number range at a resolution of 90 kHz (only 2/3 second), there is no need for such high precision because on the video decoder, the time control is only accurate to a field time either way since the VTG free-runs (or is gen-locked to something that has nothing to do with the MPEG stream being decoded).

As a result, it seems that the lower order few bits of the time-stamp going to the decoder can be discarded. In the present invention, four bits are discarded. This means that the video decoder uses 16 bits of a 20 bit number. The resolution is, thus, 5625 Hz and can represent a time difference of 11.65 seconds.

Therefore, a PAL field is 112.5 ticks of the 5625 Hz clock. An NTSC field 93.84 ticks. Hence, it is still possible to achieve timing calculations to an accuracy of about 1% of a field time which is adequate for the present invention.

#### Hardware

Figure 65 shows the hardware in accordance with the present invention. There are two modules in addition to those disclosed in Broily. The first is added just after the start code detector 201. It is responsible for generating a token. A TIME\_STAMP token occurs just before a PICTURE\_START token. In the MPEG systems stream, the time-stamp is carried in a packet header and refers to the first picture in the packet of data. Since the packets do not line up with the video data there will, in general, be the end of the previous picture before the start of the picture to which the time-stamp refers.

The time-stamp information may be supplied to the system of the present invention either via the microprocessor interface or by using a Token. In either case, the time-stamp data (16 bits) is stored in a register. A flag is set to indicate the fact that valid time-stamp information is in the register. If the data was supplied using the TIME\_STAMP token then that token is removed from the stream of tokens.

When a PICTURE\_START token is encountered, the flag that indicates the status of the register is examined. If it is clear, then no action is taken and the PICTURE\_START token and all subsequent data is uneffected. If, however, the flag indicates that valid time-stamp information is available in the register, then a TIME\_STAMP token is generated before the PICTURE\_START token. The flag is then cleared and is available for the next time-stamp that occurs.

The second hardware module is associated with the microprogrammable State Machine 218. This is simply a series of counters clocked from the 27 MHz decoder clock. The first is a prescaler that divides the clock by 4800 (the 4804.8 option shown in the diagram is discussed later). 4800 is simply 300 (27 MHz/90 kHz) times 16.

The second counter is the time counter and is incremented each time that the prescaler 278 output clock. It is reset by the reset\_time pin.

The counters in this section should probably be implemented with fully clocked feed-back flip-flops (SYNC's) which are much more resistant to a-particle corruption than the weak-feedback latches used elsewhere. (This is because of concern that the time counter in Brian might get out of step with that in the system decoder).

The microprogrammable State Machine 218 is able to read the current time indicated by the time counter and compare it with the value supplied by the TIME\_STAMP token. It can therefore determine whether it is early or late compared to the time at which it should be decoding the pictures.

The registers for use in the SCD 201 relating to time stamps are shown in Table 20.

Register name	Size/Dir.	Reset State	Description
ts_low	B/rw	-	<p>The lower eight bits of the time-stamp value.</p> <p>This register is slaved so that new values may be written into this register without affecting the value previously written (that will become part of a TIME_STAMP token).</p> <p>Writes to this register affect the master register whilst reads read-back the slave register. Until a master-to-slave transfer has been effected using ts_valid, the value written into ts_low cannot be read back.</p>
ts_high	B/rw	-	<p>The upper eight bits of the time-stamp value.</p> <p>Slaved in the same way as ts_low.</p>
ts_valid	1/w	0	<p>This bit controls the master-slave transfer of ts_low and ts_high.</p> <p>When values have been written into ts_low and ts_high the microprocessor should write the value one into this bit. It should then poll the bit until it reads back the value one. At this point, the values written into ts_low and ts_high will have been transferred into the slave registers (and can be read back) and ts_waiting will be set to one.</p> <p>The microprocessor should then write the value zero in preparation for the next access.</p>

Register name	Size/Dir.	Reset State	Description
ts_waiting	1r0	0	<p>When set to zero the registers ts_low and ts_high do not contain valid time-stamp information.</p> <p>When set to one the registers ts_low and ts_high contain valid time-stamp information. A TIME_STAMP token will be generated before the next PICTURE_START token and ts_waiting will then become zero.</p> <p>This bit should be polled to ensure that it is zero before writing a one into ts_valid to ensure that the previous time-stamp value has been used before it is overwritten by the master-to-slave transfer.</p>

Table 20. Time-stamp "SCD" registers

#### MSN Handling of Time-Stamp Information

This section details the function of the MSM 218, in accordance with the present invention, when it receives the TIME\_STAMP token.

First, a 16-bit signed time-stamp correction is added to the time-stamp that was carried by the TIME\_STAMP token. This correction is reset to zero by the MSM 218 at chip-reset and, if no action is taken, the time-stamps are unaltered. The controlling microprocessor may, however, write any value into this register to modify the time-stamp and, therefore, compensate for differential delays through the video and audio decoders.

Next, the corrected time-stamp is subtracted from the current time. The sign of this gives the direction of the error (and determines the error code, if any, generated by the MSM 218). The absolute value of the difference is then taken and the result is compared to the frame time. If the result is less than the frame time, no action is taken. As previously discussed, time can only be controlled to an accuracy of plus or minus a frame time from the nominal time because the VTG free-runs.

In the present invention, if the error exceeds a frame-time, then some correction must be made. The MSM 218 can correct the situation itself if the decoding is too early since it can simply delay the decoding until the appropriate time. However, if the decoding is later than the intended time, then this is more difficult because it is not possible to discard pictures reliably at the output of the coded data buffer. Essentially, the decoding of the sequence is broken and the most reliable way to correct the situation is to restart the decoding process in a manner similar to random-access or channel change. In order to facilitate this procedure, the control register of the MSM 218 may be programmed to discard all data until a FLUSH token is encountered.

#### Start-up

If the MSM 218, in accordance with the present invention, receives a time-stamp at a time which it recognizes as a start-up situation (e.g., after reset, following a SEQUENCE\_END token or FLUSH token and it is still before the first PICTURE\_START) then the action of the MSM 218 may be modified. If the time-stamp indicates that decoding should have occurred earlier than the current time, then the situation is handled in the same way as detailed above. However, if the time-stamp indicates that the decoding still remains to take place after the current time (which is the normal situation on start-up), then the decoder will wait until the correct time even if the error is less than one frame-time. In this way, it is possible to set the nominal decoding time as accurately as possible to the correct time. Subsequent pictures may then be decoded, up to one frame-time before or after their nominal time, without any error situation being triggered.

In addition, in the present invention the error "ERR\_TOO\_EARLY" is not generated during start-up (since it is expected that decoding would be early) irrespective of the setting of disable\_too\_early.

#### MSM Time-stamp error codes

As a result of the time-stamp handling, it is possible that one of two errors will be generated.

ERR\_TOO\_EARLY is generated if the decoding is taking place earlier than the time indicated by the time-stamp.

ERR\_TOO\_LATE is generated if the decoding is taking place later than the time indicated by the time-stamp.

ERR\_TOO\_EARLY may be suppressed, but ERR\_TOO\_LATE will always be generated unless all errors are masked out.

Table 21 describes the various time-stamp registers associated with the Microprogrammable State Machine, in accordance with the present invention.

Register name	Size/Dir.	Reset State	Description
ts_correction	16/rw	-	Correction added to each time-stamp before it is used.
frame_time	16/rw	226 or 188	Represents the tolerance on the timing of decoding pictures. Reset state determined by the PAL/NTSC pin.
time	16/ro	zero	Reset by either reset or time_reset. The current value of time.
manual_startup	1/rw	zero	When set to one, the startup is to be performed manually using decode_disable. In this case, SEQUENCE_END and FLUSH tokens at the MSM cause decode_disable to be set to one.  When set to zero, startup is performed using the time-stamp management hardware. Decode-disable is never automatically set to one.
decode_disable	1/rw	zero	When set to zero, the decoding proceeds normally.  At the start of each picture, the MSM checks the status of decode_disable and will not proceed if it is set to one.  Note that if manual start-up is to be performed (i.e., without the time-stamp management hardware) this bit should be set to one at the same time as manual-startup is set to one.
disable_too_early	1/rw	zero	When set to one, the error "ERR_TOO_EARLY" indicating that the decoding is too early is suppressed and the MSM simply waits to correct the situation.
NTSC_30	1/rw	zero	When set to one, the prescaler divides by 4804.8 rather than 4800. Set automatically when decoding 30 Hz frame rates.

Register name	Size/Dir.	Reset State	Description
discard_if_late	1/w	zero	This has no effect unless an "ERR_TOO_LATE" is generated (or would be generated if errors were not masked out). If it is set to one then data is discarded until the condition indicated by discard_until.
discard_until	2/w	0	Indicate the condition which causes time-stamp triggered discarding to be terminated.  0 - FLUSH  1 - SEQUENCE_START  2 - GROUP_START  3 - Next Picture.  Note 1 - that discarding one picture may immediately be un-done if that picture is a field picture by the generation of a dummy field to preserve the alternating top/bottom field structure. As a result if discard_until is set to "Next Picture" but the dummy field would be generated one further picture is discarded.

Table 21. Time-stamp "MSM" registers

Support for 30 Hz

The present invention does not support a 30 Hz frame rate properly. However, it will be appreciated by one of ordinary skill in the art, that the invention may decode 30 Hz data if the clock generation circuitry is modified appropriately. In this case, the system is clocked with a 27.027 MHz clock so that the typical "CCIR-601" raster produces pictures at precisely 30 Hz. In order to accommodate the 27.027 MHz clock, it must be divided by 300.3 to provide the 90 kHz clock. Since the present invention scales this value by a factor of sixteen, it is necessary to divide the clock by 4804.8.

## Introduction

This section details a Micro-codeable State Machine (MSM), in accordance with the present invention. The aim of building the MSM was to produce a machine that with small amendments can be used in a number of applications such as a VLC decoder and address generators.

The MSM of the present invention is of a general purpose nature providing support to a wide range of features. However, the underlying structure of the MSM is modular, allowing flexibility in building. Accordingly, those of ordinary skill in the art will appreciate that the present invention can be used with a variety of applications.

As shown in Figure 66, the system design is segmented into two sections. The first is a State Machine 218. This generates instructions that are passed to a data processing pipeline under the control of a two-wire interface as previously disclosed in the Broily application and incorporated by reference herein. The second section is an Arithmetic Core 219, comprising an ALU 222 and associated register file 221. This Arithmetic Core 219 is part of the data processing pipeline. It accepts data and instructions under the control of two 2-wire interfaces.\* It generates data at its output under the control of a two-wire interface. Bringing these two components together allows the definition of the complete ucode word.

## State Machine

The State Machine 218, in accordance with the present invention, provides instructions to the Arithmetic Core 219. It also provides instructions to control itself in the progression through the instructions.

The address of the instruction being passed to the Arithmetic Core 219 is held in the Program Counter. The program counter resets to 0x00 and proceeds continuously through the address. However, "jump" or "call" instructions and/or "interrupt/error" events can cause the Program Counter to reload, hence altering the order of instruction execution.

---

\* If the State Machine is also controlling upstream blocks, these two 2-wire interfaces may be combined.



The State Machine 218 allows for up to 4096 instructions in the present invention. However, it will be appreciated by those skilled in the art that other amounts of instructions may also be used and this is not intended to act as a limitation.

#### Jumps and Calls

In this implementation, all instructions are conditional jump instructions. A condition is evaluated for every instruction to determine whether or not to jump (i.e., reload the Program Counter). The two conditions "True" and "False" are provided to unconditionally jump or not jump respectively. The remainder of the conditions (sixteen in total) are based on tests on the Status bus. If the condition is not "true" or "false," the State Machine 218 will wait until the Arithmetic Core 219 has executed the instruction and fed the status bus back to the State Machine for testing against the condition. These conditions are shown below in Table 22.

Code	Condition	
0001	F	False - never jump
0010	C	Carry set
0011	NC	Carry clear
0100	Z	Zero
0101	NZ	Non-zero
0110	AN	ALU result Negative
0111	AP	ALU result Positive
1000	F	False - spare conditions
1001	F	
1010	LT	(S<V) [I-J indicates I<J]
1011	GE	~(S<V) [I-J indicates I>J]
1100	I	An index Register Incr. stepped past terminal
1101	NI	An index Register Incr. did not step past terminal

Code	Condition	
1110	V	Overflow
1111	NE	Extn bit is low

TABLE 22. State Machine conditions

If a jump is taken on an instruction with the call bit set, the next address, had the jump not been taken, will be stored as the return address. Accordingly, this forms a mechanism for routine calling. To return from the routine to the stored address, a call is made to address 0x001. Calling is only supported to a depth of one call, i.e., only one return address can be stored. Nevertheless, calling from calls, although erroneous, is not checked for in the hardware.

#### Interrupts and Errors

In the present invention, if the interrupt/error wire sampled high, an unconditional jump is made to the interrupt/error address (address 0x001). The next address that was to have been taken without the interrupt/error is stored. To return from the interrupt/error routine, is a jump to the interrupt address (0x001) is performed.

The State Machine 218, in accordance with the present invention, is hardwired to execute as either an interrupt or error routine. The difference is that interrupt routines mask out other interrupts while executing, whereas error routines do not. The State Machine 218 is currently wired as an interrupt rather than an error pin.

#### Jump Addresses

The address loaded into the Program Counter is the Jump address. The twelve bits of this address are contained in a ucode field. It can either be an absolute address or it may have portions substituted into it from the output of the ALU 222. If an address is to be substituted, the State Machine 218 will wait until the Arithmetic Core 219 has executed the instruction and fed the ALU 222 output to the State Machine for the substitution.

The format of the address, in accordance with the present invention, is shown in Table 23, "Jump Address substitution". The bits marked "a" indicate absolute address bits. The remaining address bits of lesser significance will be substituted. The LSB marked "s" is the substitute bit.

No. Bits Replaced	B	A	9	8	7	6	5	4	3	2	1	0	s
0	a	a	a	a	a	a	a	a	a	a	a	a	0
1	a	a	a	a	a	a	a	a	a	a	a	0	1
2	a	a	a	a	a	a	a	a	a	a	0	1	1
3	a	a	a	a	a	a	a	a	a	0	1	1	1
4	a	a	a	a	a	a	a	a	0	1	1	1	1
5	a	a	a	a	a	a	a	0	1	1	1	1	1
6	a	a	a	a	a	a	0	1	1	1	1	1	1
7	a	a	a	a	a	0	1	1	1	1	1	1	1
8	a	a	a	a	0	1	1	1	1	1	1	1	1
9	a	a	a	0	1	1	1	1	1	1	1	1	1
10	a	a	0	1	1	1	1	1	1	1	1	1	1
11	a	0	1	1	1	1	1	1	1	1	1	1	1
12	0	1	1	1	1	1	1	1	1	1	1	1	1
Load Return Addr.	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 23. Jump Address substitution

The address substitution feature of the present invention allows the construction of jump tables.

#### State Machine Internal Instructions

It may be desired to perform repeated conditional tests on the status bus. These instructions are internal to the State Machine 218 and require stable feedback from the Arithmetic Core 219. Therefore, these type of instructions can be marked as non-valid for

the Arithmetic Core 219, which will then fail to execute them. Accordingly, a "valid" bit is provided to mark instructions as valid for the Arithmetic Core 219.

#### State Machine Testing

In the present invention to enable the State Machine's 218 operation to be verified, a number of registers will be accessible to the microprocessor bus. Access may be gained by setting the "access" register to one and then polling the register until it reads back this value. The State Machine is then halted and it is safe to access. The machine can be restarted by writing zero to the "access" register.

When the microprocessor has access, it can read and write to the following registers:

- the program counter
- the call return address
- the interrupt return address
- the interrupt status bit (i.e., stating whether an interrupt is in progress)
- all bits of the ucode

Table 24 describes the various addresses of these registers.

The State Machine 218 can also stop itself by generating a microprocessor event. Only if the event's mask bit is set will the machine halt. Access should then be gained in the normal way when servicing this event. An event can be brought about by a call to the reset address (0x00). The call will not actually be taken, but simply generate the event after the instruction is executed. It will, nevertheless, remain at the output of the instruction ROM for inspection.

The State Machine 218 of the present invention possess a mode in which it will single step through its instructions. Single stepping is initiated by setting bit 0 of the MSSR register. The machine will then stop before each instruction. The stopped state is indicated by "1" = Stopped. The instruction about to be executed will then be at the output of the instruction ROM and is able to be changed via microprocessor access. To restart the machine, write "1" to bit 1 of the MSSR register. Both of these bit registers are synchronized and, therefore, require microprocessor access before they can be accessed.

---

## State Machine Ucode Map

Table 25 shows the microcode map for the State machine of the present invention.

Address	Use
0x000	reset address
0x001	interrupt/error address
0x002	ucode program addresses
-0xfff	

Table 25. State Machine Ucode Map

## State Machine ucode word

Similarly, Table 26 depicts the State machine microcode word, in accordance with the present invention.

Bit number	2	1	0	f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0
Bit use	a	a	a	a	a	a	a	a	a	a	a	a	s	c	Condition				v

Table 26. State Machine Ucode Word

where:

- a = address;
- s = substitute an address;
- c = call or jump;
- condition = jump condition code; and
- v = instruction valid for Arithmetic Core

### Arithmetic Core

In the present invention, the Arithmetic Core 219 performs all the data manipulation within the MSM 218. As shown in Figure 67, the general structure of the Arithmetic Core 219 includes functional blocks which select their inputs from the available buses and provide a bus as output.

The Arithmetic Core 219 is 32 bits wide, and is built from bit-slices which allows 8, 16, 24 or 32 bit data paths to be constructed in other implementations.

As depicted in Figure 68, the Arithmetic Core 219 of the invention has three main functional blocks: the token port 360, for communicating with the data stream; the ALU block 222 (and possibly other functions) for executing computations; and the Register File 221 which contains all the registers. All output buses are labeled in Figure 68. Inputs to blocks are selected from these buses. The size of these selectors and their inputs can vary and are under ucode control.

#### ALU

The ALU block 222, in accordance with the present invention, is responsible for all the computations and number manipulations in the arithmetic core. It allows quite complicated computations (such as recirculating, multiplication and division) to be performed by a combination of relatively simple operations (i.e., shifting, conditional inversion and addition). Each of these blocks is described below. Examples are then provided as to how these may be used in the Arithmetic Core 219, as a whole, to perform the more complicated computations.

#### Shift Block

In the present invention, the "shift" block allows for a 1 bit left shift, a right shift, or no shift. The 1 bit bus K is rotated into the word as if it were an extra bit. This is shown in Table 27.

ss	shift function
00	$r = l$
01	$r' = l$ ; NOP
10	$r' = (l \ll 1) + K$

ss	shift function
11	$f = (l \gg 1) + (K \ll 32)$

Table 27. Shift Block

If  $ss = 0b01$  a "NOP" is signaled to the ALU 222 as a whole. This is a No Operation and will prevent any status flags begin altered from the last operation.

#### Carry Block

The Carry block either takes the carry bit from status registers or clears it. In single word addition and subtraction operations, the carry bit will be cleared, while in multiple word operations, the carry generated by the previous operation (and stored in the status flags) will be used as the carry. This is depicted in Table 28.

c	carry function
0	$C = 0$
1	$C = H$ from status flag

Table 28. Carry Block

#### Condition Block

In accordance with the present invention, the block conditions, the Augend and the Carry to the ALU core functions are defined in Table 29.

ll	Invert function
00	$J = J$ $C' = C$
01	$J' = \sim J$ $C' = \sim C$
10	$J' = J \& L$ $C' = C \& L$

#	Invert function
11	$J' = (L \ ? \ J; \sim J)$ $C' = (L \ ? \ C; \sim C)$

Table 29. Condition Block

ALU Core

The ALU core 222 of the present invention performs a simple set of logic and arithmetic functions using two's complement arithmetic. These are defined in Table 30.

#	ALU core functions	
0	$F + J' + C'$	Add
1	$F \wedge J'$	XOR
10	$F \& J'$	AND
11	$F   J'$	OR

Table 30. ALU Core

From the result of the ALU core 222, four status flags are generated. (See Table 31.) These are both stored in the Register File 221 (as shown in Table 35) and are sent back to the State Machine 218 for comparison with condition codes.

Meaning	Invert function
Carry	Carry Out from ALU operation
Zero	ALU result is zero
Negative	MSB of ALU result = 1
Overflow	ALU operation overflows

Table 31. Status Flags generated by the ALU core



ALU Ucode Word

Table 32 illustrates the ALU microcode word.

Bit number	6	5	4	3	2	1	0
Bit use	s	s	i	i	f	f	c

Table 32. ALU microcode word

where

ss is the shift block controls  
 ii is the condition block controls  
 ff is the ALU core controls  
 c is the carry block controls

Use of the ALU

Table 33 describes the bits patterns for the various functions of the ALU, in accordance with the present invention.

Bit number	6	5	4	3	2	1	0
Addition (I-J)	0	0	0	0	0	0	0
Subtraction (I-J)	0	0	0	1	0	0	0
Multiplication	1	0	1	0	0	0	0
Division	1	0	1	1	0	0	0

Table 33.

Register File

Figure 69 illustrates the register file 221 of the present invention. The register file 221 contains 64 thirty-two bit word registers. The register file 221 can address partial words, i.e., the file can be addressed as 64 x 32 bit, 128 x 16 bit, 256 x 8 bit, 512 x 4 bit, 1024 x 2 bit, or 2048 x 1 bit formats. The address is provided directly from the ucode or the

address may have portions of it substituted into from special registers. This allows indexed access of the register.

At each instruction, a read-modify-write will be done on a single register. The read-modify-write facilitates the writing of partial words back into the file. The source of the write is determined by an external multiplexer with its own independent ucode. If no write is desired, the output of the register file 221 should be selected by the multiplexer.

Partial words will be treated as signed or unsigned numbers dependent on bit 0 of the mode register. If the partial word is negative (i.e., it has its MSB set) it will be sign extended up the full width of the bus. This allows the easy use of partial words in arithmetic.

Three locations in the register file 221 of the present invention are also connected to a dedicated bus, but they are still allowed to be used in parallel with other register file locations. These are the A and B registers and the status register shown in Figure 69. The register file also contains the index registers for address substitution with accompanying terminal count registers, constant registers and a mode register specifying modes of the register file.

#### Register File Addressing

The addressing, in accordance with the present invention, must cope with two different features: variable length addresses for accessing varying width portions of words, and address substitution.

To address partial words requires a longer address. Therefore, all addresses are of variable length and they are encoded as follows: where "a" is an address bit, the least significant of the address bits is "S", the substitution bit.

Data Width	B	A	9	8	7	6	5	4	3	2	1	0	S
1	1	a	a	a	a	a	a	a	a	a	a	a	a
2	0	1	a	a	a	a	a	a	a	a	a	a	a
4	0	0	1	a	a	a	a	a	a	a	a	a	a
8	0	0	0	1	a	a	a	a	a	a	a	a	a
16	0	0	0	0	1	a	a	a	a	a	a	a	a
32 (24)	0	0	0	0	0	1	a	a	a	a	a	a	a

TABLE 34. Variable width addressing

The addressing is big endian. That is to say the higher, more significant portions of the words are addressed with lower addresses.

Portions of the addresses "a...a" can be substituted with one of the index registers. Using an address of an eight bit word as defined in Table 34 as an example, Table 35 shows how to define the number of least significant bits which are to be substituted. All trailing zeros are substituted.

Bits to be substituted	C	B	A	9	8	7	6	5	4	3	2	1	0	S
0	0	0	0	1	a	a	a	a	a	a	a	a	a	0
1	0	0	0	1	a	a	a	a	a	a	a	a	0	1
2	0	0	0	1	a	a	a	a	a	a	a	0	1	1
3	0	0	0	1	a	a	a	a	a	a	0	1	1	1
4	0	0	0	1	a	a	a	a	a	0	1	1	1	1
5	0	0	0	1	a	a	a	a	0	1	1	1	1	1
6	0	0	0	1	a	a	a	0	1	1	1	1	1	1
7	0	0	0	1	a	a	0	1	1	1	1	1	1	1
8	0	0	0	1	a	0	1	1	1	1	1	1	1	1

TABLE 35. Address substitution

For example, substituting 4 bits into a thirty-two bit address would have the form 0b000001aaa011111 or substituting zero bits into a one bit address 0b1aaaaa00000000.

In the invention, the substitution will come from one of the two eight bit index registers, which is specified in the register files ucode word. It can, therefore, be seen that a maximum of eight bits can be substituted into an address.

It can also be seen that with the above scheme, it is possible to use illegal addresses, such as 0b000000000000000 or 0b111111111111111. Illegal addresses will result in no address being accessed, leaving the output bus of the register file unknown.

#### Register File Register Types

In the present invention, there are a plurality of register file register types. Each is described as follows:

- Independently bused registers

Three registers (A, B, and Status register) have their own dedicated buses, as well as being accessible in the normal way in the register file. This allows the registers to be

connected to more places in the arithmetic core 219 and to be accessible in parallel to others in the register file. The independent buses can only access the registers in their full width, i.e., 32 bits wide.

There is no ucode write enable to these registers. Writing to them is only by way of an external multiplexer which has its own ucode control word. To prevent a write, they must be written to with their own value as shown in Figure 70.

When the independent bus registers are written to as if in the register file, the independent bus write is suppressed.

The Status register is implemented as an independent bus register. The bits of the register are defined in Table 36.

Bit		Meaning	Comment
0	1	Index Reg	An index register increments passed its terminal count
1	E	Extn	Extension bit from input
2	V	Overflow	ALU operation overflows
3	N	Negative	MSB of ALU result = 1
4	Z	Zero	ALU result is zero
5	C	Carry	Carry from ALU operation
6		Gnd	Unused
7		Gnd	Unused

TABLE 36. Definition of the Status register

- Index and terminal count registers •

Two eight bit Index registers are provided for substituting into addresses. One of these can be incremented per instruction under the control of the ucode. Furthermore, each is accompanied by a terminal count register. When the register incremented is passed, its terminal count will be reset to zero.

The index registers are called Y and Z which have terminal count registers U and V, respectively. All of these can be accessed in the register file.

Index register Z has a predefined decoder attached to its output (at present this decode is an inversion). Dependent on the Index\_Mode in the mode register (bit 1), this decoder rather than the index registers, will be used in address substitution and read from Z in the register file. (Index\_Mode = 1 read decode, Index\_Mode = 0 read count)

• Constant registers

In the present invention, sixteen of the register file's 32 bit locations will be predefined constants. These may be read out as normal registers. Writing to these locations will have no effect. (The constants selected for the current embodiment are 0-7. However, it will be appreciated that other numbers of constants may be used.)

This implementation of constants in accordance with the present invention, does away with the need for a constant field in the ucode and for a constant bus in the arithmetic core. It does, however, limit the number of constants useable in the program. (The number 16 is negotiable.) These constants are programmed on a per instance basis. Furthermore, very frequently used values could be connected to multiplexers, if necessary.

Register File Address Map

Table 37 shows the Register File address map for the present invention.

32 Bit Location	Bits	Register
0x00	All	A register
0x01	All	B register
0x02	7:0	Status register
0x02	8	Sign Extend mode
0x02	9	Index Decode mode
0x02	31:10	Normal register
0x03	7:0	Y index register
0x03	15:8	Z index register

32 Bit Location	Bits	Register
0x03	31:16	Normal register
0x04	7:0	U terminal count register
0x04	15:8	V terminal count register
0x04	31:16	Normal register
0x05-0x37	All	Normal registers
0x37-0x3F	All	Constants

TABLE 37. Register File Address Map

**Register File Ucode Word**

Table 38 shows the Register File microcode word for the present invention.

Bit No.	d	c	b	a	9	8	7	6	5	4	3	2	1	0
Bit use	a	a	a	a	a	a	a	a	a	a	s	r	i	

Table 38. Register File Ucode Word

where

- a = whole register file address (always 12 bits)
- s = substitute bit
- r = index register to use for substitution: select Y, Z index registers if n = 0, 1 respectively
- i = increment index register specified by r

**Token Port**

The Token Port of the present invention is the arithmetic core's connection to the data stream. It is a two-wire interface connection.

The data at the Token Port input is only defined during a Token Port reading cycle. It should, therefore, be used during read cycles only.

If the input port does not contain valid data during a read cycle or the output port is not accepting during a write cycle, the Arithmetic Core will stall. Accordingly, it will perform no operation, read no new ucode word, and write no registers. It will only restart when these conditions do not exist.

Token Port Ucode Word

Table 39 depicts the Token Port microcode word.

Bit No.	1	0
Bit use	1	0

Table 39. Token Port Ucode Word



where

- I = read into input port  
O = write from output port

#### Multiplexers

The selection of sources for blocks is done by the use of multiplexers. Almost all combinations of bus are allowed (with the exception that the Input to functional blocks, e.g., the ALU must be from storage blocks, e.g., Token Port or Register File).

The multiplexers are either 2, 4 or 8 input. They, therefore, use 1, 2, or 3 bits of ucode word, respectively, to control their selection of inputs.

#### UPI Memory Map

Table 40 shows the MSM address map, in accordance with the present invention.

Address	Bits	Location
0x000	0	MSM Event bit
0x001	0	MSM Mask bit
0x100	7	Access bit
0x101	0	MSSR Set single stepping
0x101	1	MSSR Monitor Single Stepping
0x101	2	MSSR Interrupt status register (Read Only)
0x102	3:0	Program Counter MSB
0x103	7:0	Program Counter LSB
0x104	3:0	Call Return Address MSB
0x105	7:0	Call Return Address LSB
0x106	3:0	Interrupt Return Address
0x107	7:0	Interrupt Return Address

Address	Bits	Location
0x200 - 0x2ff	7:0	Register File

Table 40. MSM Address Map

## Introduction

In the MPEG coding standards (both EGMP-1 and MPEG-2) the quantized coefficients are coded as "events". Each event is coded as a RUN and a LEVEL. The RUN is the number of zero coefficients that precede a given non-zero coefficient. The LEVEL is the value of that coefficient. In addition, one special event, End-of-block, is used after the last non-zero coefficient to indicate that the remainder of the block is all-zeros.

For example, assume the following sequence of coefficients:

1, -7, 0, 3, 0, 0, 0, -1, 0, 0, 0, ...0 (total of 64 coefficients)

These would be modeled by the following events represented as (RUN, LEVEL):

(0, 1) (0, -7) (1, 3) (3, -1) (EOB)

It is the task of the inverse modeler to reverse the modeling process such that each of the 64 coefficients is represented as a simple number for subsequent processing.

## Interfaces

The following signal pins are used to transfer data into the inverse modeler of the present invention:

- level [11:0]
- run[5:0]
- ln\_extn
- ln\_valid
- ln\_accept

Tokens are transferred on the level[11:0] bus (in the lower order eight bits; level[7:0]).

run[5:0] serves as an auxiliary bus to carry the RUN information. It has no meaning except in the data words of a DATA token.

The following signals are used at the output of the inverse modeler:

- out\_data[11:0]

- out\_extn
- out\_valid
- out\_accept

### Functional Description

Data in DATA tokens is expanded so that there is always 64 coefficients in the DATA token presented at the output of the inverse modeller. In most cases, the last data word of the DATA token will not cause the 64<sup>th</sup> coefficient to be generated. This is not an error, it is just that at this point the EOB event would have been coded in the bitstream. Therefore, in this situation the inverse modeller must continue to output zero data token words until a total of 64 coefficients have been produced at the output.

In certain circumstances (e.g., when a data error occurs) it is possible for the DATA token at the input to the inverse modeller to represent more than 64 coefficients. In this situation, the modeller must discard all the extra data and produce a token at its output that contains just 64 coefficients.

All non-DATA tokens that appear at the input are simply transferred, unmodified, to the output of the inverse modeller.

### Timing Requirements

It is a requirement of the present invention that data flow through the inverse modeller at the clock rate.

In the situation where there are no gaps at the input to the Imodel and the circuitry connected to the output does not cause the Imodel to stall (i.e., in\_valid = 1, out\_accept = 1) then a new data word will appear at the output of the Imodel every clock cycle. Note, however, that in this situation, the Imodel may not accept new data at its input on every single clock cycle because a non-zero RUN (in a DATA token) will cause more than one data word to be produced for each input.

### Microprocessor Interface Access

The inverse modeller circuitry of the present invention is not required to be connected to the MPI in its normal mode of operation. Note that the error condition (too many coefficients) should *not* produce a microprocessor interrupt. It is simply dealt with internally by discarding the extra data.

However, microprocessor access will be required for the snooper (test) circuitry at the input of the block.

## Introduction

In the MPEG coding standards, the coefficients are "zig-zag" scanned so that the lower frequency coefficients are transmitted before the higher frequency coefficients.

It is the function of the inverse zig-zag, in accordance with the present invention, to convert the one-dimensional stream of coefficients it receives from the inverse modeller into a two-dimensional array of coefficients that can be processed by the IDCT.

In MPEG-1, only one scan path was used, this was literally a zig-zag (hence, the name), MPEG-2, however, uses two scan paths. The first is the original MPEG-1 path, the second is optimized for use in interlaced coding where there tends to be unusually large vertical frequency components.

In addition to the coefficients which are obviously transmitted in zig-zag scan order, the quantization matrices are downloaded in zig-zag scan order as well. This occurs in MPEG-1, H.261 and JPEG. As a result, the present invention has its quantizer before the inverse zig-zag (which was implemented as part of the IDCT). The quantizer, therefore, operates on a one-dimensional stream of coefficients which arrive in the same order as the downloaded quantization matrix coefficients. Hence, the quantizer simply has to associate the first coefficient with the first matrix element, the second coefficient with the second matrix element, and so forth.

However, since there are now two scan paths in MPEG-2, a new approach was taken on the present invention in which the inverse zig-zag precedes the inverse quantizer. Both the coefficients and the downloaded matrices are inverse scanned and the inverse quantizer now operates on the two dimensional data. It should be noted that this is only possible because in all three representations of the data (two zig-zag scans and the raster-scan order at the output of the IZZ) the first coefficient is always first and the last coefficient is always last. The first coefficient is specially treated in the Iquant because it is the DC term. The last coefficient is specially treated because it may need to be modified as a result of mis-match control as a function of the values of all the other coefficients (so it must be last!). The 62 remaining coefficients are all handled in the same manner (excepting that each has its own quantization matrix element).

## Interfaces

The following signals are used at the input of the inverse zig-zag of the invention:

- in\_data[11:0]
- in\_extn
- in\_invalid
- in\_accept

The following signals are used at the output of the inverse zig-zag:

- out\_data[11:0]
- out\_extn
- in\_invalid
- out\_accept

### Functional Description

The IZZ responds to the following tokens:

- PICTURE\_START
- ALTERNATE\_SCAN
- DATA
- QUANT\_TABLE

All other tokens are passed, unmodified, through the IZZ.

The PICTURE\_START token causes the IZZ to reset its internal state which represents which of the two scan paths is in force (e.g., alternate\_scan) to zero (indicating the MPEG-1 scan).

ALTERNATE\_SCAN is a token which can be allocated the value 0xe5 with a mask 0xe. The ALTERNATE\_SCAN token is shown in Table 41.

E	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	1	s

Table 41. Alternate\_Scan Token

"s" is the indication of which scan to use for subsequent DATA tokens and is, therefore, loaded into the IZZ register "alternate\_scan".

DATA tokens are re-ordered according to scan path zero (the MPEG-1 scan path) irrespective of the setting of `alternate_scan`. Note that `alternate_scan` must retain whatever value it had (i.e., must not be set to zero) so that subsequent DATA tokens are correctly handled.

QUANT\_TABLE tokens are re-ordered according to scan path zero (the MPEG-1 scan path) irrespective of the setting of `alternate_scan`. Note that `alternate_scan` must retain whatever value it had (i.e., must not be set to zero) so that subsequent DATA tokens are correctly handled.

#### Mal-formed Tokens

Both the DATA and QUANT\_TABLE tokens may be mal-formed. Clearly, the DATA token should be correct since the model should have ensured that it is correctly formed. However, no such assurance is available for QUANT\_TABLE. Since handling the mal-formed QUANT\_TABLE tokens must be implemented, it should also be implemented for DATA tokens as well.

In accordance with the present invention, DATA and QUANT\_TABLE tokens are too short when they appear at the input to the IZZ should result in a token at the output with the correct number (64) of data words. The data contained in those words is unimportant and will probably be whatever junk happened to be in the re-ordering RAM before the start of the token. Similarly, DATA and QUANT\_TABLE tokens that are too long should also result in correctly formed tokens at the output. The first 64 coefficients (matrix elements) should be used, the remainder should be discarded.

Following a mal-formed token, all subsequent (correctly formed) tokens should be handled properly.

There is no requirement for a microprocessor interface error (interrupt) to be generated.

#### Raster Scan Order

At the output of the IZZ, the DATA and QUANT\_TABLE tokens of the present invention represent two-dimensional data. However, the coefficients are still actually transferred as a one-dimensional series of numbers. The question arises whether the data should be transferred as rows or as columns.



The prediction circuitry will require the pet-domain data to be organized in raster-scan order. Since the IDCT transposes the data it follows that the data going into the IDCT must be the other way around. Table 42 illustrates the order of the coefficients transferred at the output of the IZZ for DATA and QUANT\_TABLE tokens.

		increasing horizontal frequency →							
		0	1	2	3	4	5	6	7
0		0	8	16	24	32	40	48	56
1		1	9	17	25	33	41	49	57
2		2	10	18	26	34	42	50	58
3		3	11	19	27	35	43	51	59
4		4	12	20	28	36	44	52	60
5		5	13	21	29	37	45	53	61
6		6	14	22	30	38	46	54	62
7		7	15	23	31	39	47	55	63

Table 42. IZZ Output Coefficients

#### Microprocessor Interface Access

There is no requirement for microprocessor access in the normal functioning of the IZZ. However, access will probably be required so that the reordering RAM can be tested. It is also expected that there will be no requirement for a snooper. The one at the start of the lmodel is sufficient for both blocks.

## Introduction

This section deal with predictions. In this introductory section, all possible prediction modes are enumerated and diagrams are provided for each one to explain exactly what must be done.

Throughout this section no special attention is given to operations, such as half-pel filtering, that occur in the horizontal dimension. This is because these operations are the same as those on Brollly. In the vertical dimension, however, things are very different because of the interlaced picture format.

### Prediction in Frame Pictures

#### Frame-based Prediction

In this mode, a prediction is formed from a reference frame. The result is as if the two reference fields were first combined into a frame and then a prediction were made from that frame. Note that this is precisely the situation as described in Brollly.

Half-pel filtering may be made in the vertical direction and this is triggered by the least significant bit of the vector. In addition to the least significant bit, the next most significant bit (bit 1) has special significance since this will determine whether the top line of the prediction comes from the top reference field or the bottom reference field.

Thus, four cases have to be considered, each dependant upon the binary value of the least significant two bits of the vertical vector

$$\text{vector}[1] = 0, \text{vector}[0] = 0$$

As shown in Figure 71, just 16 lines (8 for the chroma) are read (since there is no half pel filtering. 8(4) lines from each reference field.

$$\text{vector}[1] = 0, \text{vector}[0] = 1$$

Likewise, as shown in Figure 72, 17(9) lines are read, 9(5) lines are read from the top reference field, 8(4) lines from the bottom reference field.

$$\text{vector}[1] = 1, \text{vector}[0] = 0$$

Again, as shown in Figure 73, just 16(B) lines are read but note that now the top line of the prediction has been read from the bottom reference field.

vector[1] = 1, vector[0] = 1

And, Figure 74 shows 17(B) lines are read, 8(4) lines are read from the top reference field, 9(5) lines from the bottom reference field.

Accordingly, bit 1 indicates which reference field holds the top-most line that must be read to produce the prediction. In addition, if bit 0 is also set, it indicates which reference field has the extra line to enable half-pel filtering to be performed.

It is clear that half-pel prediction cannot be performed until both fields have been read from DRAM.

Great care must also be taken when scaling vertical motion vectors to obtain offsets in the field store. The following table, Table 43, illustrates the effect:

Vector	Bit pattern	Offset in field	
		top field	bottom field
-2	...11100	...11110 (-2)	...1111 (-2)
-1.5	...11101	...11111 (-1)	...11110 (-2)
-1	...11110	...1111 (-1)	...11111 (-1)
-0.5	...11111	...00000 (0)	...11111 (-1)
0	...00000	...00000 (0)	...00000 (0)
0.5	...00001	...00001 (1)	...00000 (0)
1	...00010	...00001 (1)	...00001 (1)
1.5	...00011	...00010 (2)	...00001 (1)
2	...00100	...00010 (2)	...00010 (2)

Table 43

#### Field-based Prediction (in a frame picture)

In this mode, each field is treated independently. A separate vector is used for each of the two fields. Associated with each vector is an additional single bit flag (`motion_vertical_field_select`) that indicates whether prediction should be made from the top reference field or the bottom reference field. The bottom bit of the vector still indicates the need for half-pel filtering, but bit 1 has no special significance. Note that a field vector measures different units of a frame vector; a field vector with the value  $n$  represents the same actual displacement (on the pless) as a frame vector with the value  $2n$ .

This time, however, there are sixteen cases to consider (since there are four binary variables; `motion_vertical_field_select` for each of the two vectors and bit 0 for each of the two vectors). There are too many cases to draw, hence, the following figures only deal with the prediction of the top field. The bottom field is obtained in an analogous manner.

As depicted in Figure 75, `motion_vertical_field_select = 0`, `vector[0] = 0`

8(4) lines are read from the top reference field to form the top field of the prediction.

Figure 76 shows `motion_vertical_field_select = 0`, `vector[0] = 1`

9(5) lines are read from the top reference field which are then half-pel filtered to form the top field of the prediction.

Likewise, Figure 77 depicts `motion_vertical_field_select = 1`, `vector[0] = 0`

8(4) lines are read from the bottom reference field to form the top field of the prediction.

And, Figure 78 illustrates `motion_vertical_field_select = 1`, `vector[0] = 1`

9(5) lines are read from the bottom reference field which are then half-pel filtered to form the top field of the prediction.

#### Dual Prime (in frame pictures)

Dual prime is a special case of the Field-based prediction of the previous section. Essentially, dual-prime combines two features:

- A special method of coding the vectors so that despite the fact that four independent field predictions are formed (independent in the sense that they

each have a distinct vector) *effectively* only one motion vector is transmitted. Thus, the vector overhead is dramatically reduced.

- For each field, the prediction information is read from each of the reference fields. This is then averaged to form the final prediction. This is very similar to the B-picture case when a separate forward and backward prediction is made and then averaged.

In the present invention, the vector decoding will all be performed in the parser. Accordingly, when the prediction circuitry receives data, there really will be four separate vectors.

The dual-prime averaging will be performed by re-using the B-frame averaging circuitry (dual-prime cannot itself be used in a B-frame). Hence, the only associated complication for the prediction circuitry is involved in the signaling that indicates that the backwards predictions (using backwards vector tokens etc.) should be performed from the *forward* reference fields (as opposed to the backward reference fields). Since a P-picture should never normally request a backward prediction, the prediction circuitry merely needs to keep a record of the picture type (P or B) in order to be able to decide which reference store to use for a "backward" prediction.

#### Prediction in Field Pictures

##### Field-based Prediction

This is very similar to field-based prediction in frame pictures. There are four cases depending on `motion_vertical_field_select` and the least significant bit of the motion vector. Note that it is not really relevant to discuss top-fields and bottom-fields in the prediction that is formed, since the prediction is simply for the picture being decoded (which is either all top-field or all bottom-field).

Figure 79 illustrates `motion_vertical_field_select = 0`, `vector[0] = 0`

16(8) lines are read from the top reference field to form the prediction.

Figure 80 shows `motion_vertical_field_select = 0`, `vector[0] = 1`

17(9) lines are read from the top reference field and half-pel filtered to give the prediction.

Figure 81 depicts  $\text{motion\_vertical\_field\_select} = 1, \text{vector}[0] = 0$

15(8) lines are read from the bottom reference field to form the prediction.

Figure 82 shows  $\text{motion\_vertical\_field\_select} = 0, \text{vector}[0] = 1$

17(9) lines are read from the bottom reference field and half-pel filtered to give the prediction.

#### 16x8 MC

In this mode, the macroblock is divided into two 16x8 regions, one above the other. For each region, a separate field vector is transmitted. Again, there are sixteen cases to consider (since there are four binary variables:  $\text{motion\_vertical\_field\_select}$  for each of the two vectors and bit 0 for each of the two vectors). Again, there are too many cases to illustrate so the following figures need only deal with the upper 16x8 region. The lower region is obtained in an analogous manner.

Figure 83 shows  $\text{motion\_vertical\_field\_select} = 0, \text{vector}[0] = 0$

8(4) lines are read from the top reference field to form the prediction of the upper 16x8 region.

Figure 84 depicts  $\text{motion\_vertical\_field\_select} = 0, \text{vector}[0] = 1$

9(5) lines are read from the top reference field and half-pel filtered to form the prediction of the upper 16x8 region.

Figure 85 illustrates  $\text{motion\_vertical\_field\_select} = 1, \text{vector}[0] = 0$

8(4) Lines are read from the bottom reference field to form the prediction of the upper 16x8 region.

Figure 86 depicts  $\text{motion\_vertical\_field\_select} = 1, \text{vector}[0] = 1$

9(5) lines are read from the bottom reference field and half-pel filtered to form the prediction of the upper 16x8 region.

Dual Prime in Field Pictures

Dual prime in a field picture is simply a special case of field prediction in a field picture. Two field vectors will be used (one will refer to the top reference field, on to the bottom reference field and the Parser will ensure this). One of the predictions will appear to be making a backwards prediction, but because this is a P-picture, the prediction circuitry will interpret this as a second forward prediction. The two resulting predictions will then be averaged using the same circuitry as that used for B-frame averaging.

### Overall organization

Figure 67 shows the overall organization of the display pipeline, in accordance with the present invention. Data arrives from the DRAM interface on a single multiplexed interface. Moreover, the DRAM interface will supply data in lines that are rounded up to the next 32 byte boundary above the correct number of bytes. However, the pets toward the end of the line that may lie outside the intended display area.

In addition to the data, the DRAM interface will supply one bit for each channel (Y, Cr and Cb) that indicates whether the byte is the last in the current display line. A further bit is supplied that indicates which field the data comes from.

The first block in the display pipeline of the present invention splits apart the three channels. Chrominance (Cr and Cb) data is supplied to the vertical upsamplers 210. Luminance (Y) data can be delayed in a FIFO if desired.

The vertical upsamplers 210 have the task of upsampling the chrominance data by a 2:1 factor so that there are as many lines of chrominance data as there are of luminance data. In order to do this the vertical upsamplers store each line of chrominance data and produce output pets that are interpolated between this line and the subsequent line.

The next stage in the display pipeline is labeled "Horizontal Align 370". This is implemented as part of the horizontal upsampler 212. Its task is to align the data so that at the start of each line, the first pet of each of the three channels is supplied to the horizontal upsampler 212 correctly. At the end of each line, it is expected that, in general, the channels will "run out of data" at different times. The "Horizontal Align" block 370 has the job of discarding this extra data from the channels that have too much data while stalling the other channels so that they wait until all three channels are aligned and ready to commence the next display line.

In the invention, the horizontal upsamplers 212 upsample the data horizontally to stretch the data to fill the glass of a TV screen. In order to save silicon area, the filter is shared between the three channels. This can be done because the total output rate of the filter must be 27 Mbytes/s (the clock rate). The data is multiplexed in the CCIR 601 order so that the data stream produced is simply multiplexed into the final data stream.

Note that the horizontal upsamplers 212 merely take the amount of data supplied by the DRAM interface and scale it by a selected factor. In general, they will produce too little or



too much data for the actual line length in the raster. This is handled in the output multiplex.

Also, note that the "Horizontal Align" block 370 does not need to know how many pels of each channel will be required to complete the line. It is very difficult to calculate this number because the relation between the number of input pels to output pels for the upsampling filter is not very simple. The Horizontal Align block 370 simply supplies data to the horizontal upsampler 212 on each of its three channels "on demand," i.e., the horizontal upsampler "pulls" the required number of pels into it in the required order. At the end of the display line, one of the channels will run out of data first and this indicates that the remaining data for the other channels (if any) should be discarded.

The VTG 333 simply counts through the raster and produces a series of timing signals that are supplied to the output multiplex 371. Some of these signals are internal signals which tell the output multiplex 371 how to build the final raster. Other signals are "external" signals, such as sync and blanking, and these are also supplied to the output multiplex 371 circuitry so that they are delayed by the same number of clock cycles as the data.

The output multiplex 371 block has several tasks. The most interesting of these is probably the task of removing the two wire "interfereceness" from the data. Data supplied from the horizontal upsampler 212 still has an associated valid signal (and the output multiplex provides an accept signal). Data at the output of the multiplex has no two-wire interface, it is simply clocked out, one byte per clock cycle.

The output multiplex 371 also has the job of painting a border around the picture. The top and left borders are painted under the control of the VTG 333. The VTG 333 simply tells the output multiplex 371 to produce the requisite number of pels of border color. At the right and bottom of the picture, the output multiplex 371 paints its own border, i.e., it knows to do this because it runs out of picture data.

The final block in the display pipeline is the 8-bit to 16-bit output mode converter 372. This is quite simply a flip-flop and a multiplexer. It is intended that this be implemented at the output PAD itself. By doing this, it is possible to simply route an 8-bit bus, rather than a 16-bit bus. Each bit will go to two output pads.

## Horizontal Upsampler

### Introduction

In accordance with the invention, the Horizontal Upsampler 212 performs the task of upsampling or interpolating the decoded picture in order to stretch it to fit the display raster.

The upsampler 212 of the present invention can operate in four modes:

- 1) 1:1 - Output is the same as the input
- 2) 2:1
- 3) 3:2
- 4) 4:3

After some picture simulations and consideration of likely implementation costs, it has been decided to use a three tap filter to perform the interpolation.

The filter is a "polyphase" filter in the sense that each successive output is generated using a different set of filter coefficients. The number of phases is always equal to the numerator of the upsampling ratio. Thus, the 4:3 upsampler has four phases, every fourth output sample being generated using the same filter coefficients.

Since the upsampler 212 is generating more output data than it accepts as input data, it is clear that a new input sample is not accepted on every clock cycle. In fact, the number of phases on which the filter does not accept new input is the difference between the numerator and the denominator of the upsampling ratio. In each of the ratios (except 1:1) this is one. Therefore, for each complete cycle around the phases, on one of the phases no new input data is accepted. In this case the data is the same as for the previous phase. The filter coefficients are, however, different to the previous phase.

### 4.3 Upsampling

In 4:3 Upsampling, the filter coefficients are shown in Table 44 while Figure 88 shows the filter in operation. The output pels are essentially formed as weighted averages of the input pels.

Phase	C[0]	C[1]	C[2]
0	0	356	0
1	42	220	-6
2	128	128	0
3	-6	220	42

Table 44. 4:3 Filter Coefficients

Note that no new input data is accepted before the final phase (phase 3) is calculated.

### 3:2 Upsampling

Table 45 illustrates 3:2 upsampling, while Figure 89 illustrates filter operation.

Phase	C[0]	C[1]	C[2]
0	0	256	0
1	68	194	-6
2	-6	194	68

Table 45. 3:2 Filter Coefficients

### 2:1 Upsampling

Likewise, Table 46 illustrates 2:1 upsampling and Figure 90 shows filtering thereof.

Phase	C[0]	C[1]	C[2]
0	0	256	0
1	0	128	128

Table 46. 2:1 Filter Coefficients

Note that Phase 1 could equally well have been described as having filter coefficients, 128, 128, 0. This has the advantage that the filter coefficients would then be the same as

for Phase 2 of the 4:3 upsampler. However, it has the disadvantage that the rule "no new input is accepted when computing the last phase" would not be true.

#### Boundary Effects

At the edge of the picture, it is necessary to produce output pels that are formed from pels that lie outside the picture area. In order to avoid this problem, it is necessary to pixel-repeat edge pels so that the filter may proceed without realizing that it is at the edge of the picture.

In the case of a three tap filter, as in the present invention, it is necessary to repeat just one pel at the left of the image and one at the right. (A five tap filter would have required 2 at the left, 2 at the right). This is shown in Figure 91.

Conceptually, therefore, the implementation could be viewed as being formed of two boxes:

Note the scheme doesn't really work properly in the case that the picture is not a multiple of 16 pels wide because the DRAM interface will always supply data that is a multiple of 16 pels wide.

However, although this problem is known, we are not going to do anything about it. Most picture are multiples of 16 pels wide anyway, and in any case it is only the very last pel of the line that can be affected by the boundary effect. This is illustrated in Figure 92.

#### The Number of Output Pels

In the present invention, the upsampler will produce a defined number of output pels for a given number of input pels. This is important because this allows the parser State Machine to decide how many pels will be produced at the output of the upsampler and, hence, how many pels need to be cropped (or border pels added) in order for the picture to fit into the raster.

The first valid output from the horizontal upsampler should occur in response to the third pel being input to the upsampler (since this is a three tap filter). Since one pel is repeated, this will occur when the second actual pel is input to the upsampler.

The last valid output should occur when all of the possible output samples have been produced in response to the last (i.e., repeated) pel being input. Since the last phase of the poly-phase filters is computed using the same input data as the second-last phase, it

is possible that either one or two output pels are produced as a result of this last repeated pel entering the upsampler.

If this is done, the upsampler will produce "q" output samples:

EQ 1.

$$q = N(pDIVM) + (pREMM)$$

in response to "p" input samples for an N:M upsampler.

For example, for a 4:3 upsampler, Table 47 could be drawn up as follows:

p (input pels)	q (output pels)
1	1
2	2
3	4
4	5
5	6
6	8

Table 47. Number of Output Pels for 4:3 Upsampler

#### Position Signals

Two signals are transferred along with the video data in the present invention. They allow the output multiplex to ensure that the data is printed into the appropriate position in the output raster. These are:

- last\_in\_line
- field\_id

last\_in\_line is active for one pel time and signals that the associated pel is the last pel in a scan line.

field\_id indicates which field the data belongs to. "0" indicates the spatially upper field, "1" indicates the spatially lower field. Note that this designation applies before any border lines and the like, are applied to the decoded image. field\_id changes state one pel too early, i.e., between the second to last and the last pel of the field. This allows the last pel of the field to be identified without waiting for the first pel of the next field. However, there may be no "next field" if decoding stops for some reason. The field\_id signal is shown in Figure 93.

If a true field indicator is required, it can be obtained by delaying field\_id by one pel time.

Since these signals work their way along side the data through the entire display pipeline, it is important to use two signals, not three (which would allow a last pel in field signal) because it saves many flip-flops.

#### Multiplexed data

When position signals are applied to multiplexed data, care needs to be taken.

The data is multiplexed in the order,  $C_2 \vee C_1 \vee$ .

In the present invention, the three samples ( $C_2 \vee C_1$ ) are co-incident in time and should, therefore, be viewed as indivisible. The remaining byte ( $\vee$ ) is positioned between the preceding ( $C_2 \vee C_1$ ) pel and the subsequent ( $C_2 \vee C_1$ ) pel.

As a result, the last byte in the line will either be the  $C_1$  or  $\vee$ . (Note that upsampling by 3:2 may produce an odd number of Y pels.) If the last byte in the lines is  $C_1$ , then, there should be a discontinuity in the multiplex signal because the first byte of a line is always  $C_2$ :

$$(C_2 Y_1 C_1 X_1)(C_2 Y_1 C_1)(C_2 Y_1 C_1 X_1)(C_2 Y_1 C_1)$$

#### Horizontal Alignment

At the input of the upsampler, there is no guarantee that the three different channels will line up.

In order to achieve alignment, in the present invention, a "protocol" between the horizontal upsampler and the horizontal alignment blocks needs to be agreed. In accordance with the present invention, the protocol performs as follows:

- The horizontal block supplies pels, on demand, to the horizontal upsampler. When it runs out of data for a given channel, it will signal this to the filter using a signal marking the last pel of the line. This will only happen for the repeated pel.
- The horizontal upsampler ensures that once it has been supplied the last pel from a given channel it will not ask for another pel from that channel in the current line. However, the filter continues to operate, taking any necessary pels from other channels, until just before it will demand a pel from the channel that it knows has run out of data. The filter marks the last pel it can produce as the output as the last in the line. At this point, it resets itself as ready for the next line of data.
- When the horizontal upsampler sees the filter accept data for a channel that has already been exhausted, it knows that the filter is asking for the first pel of the next line. At this point, any remaining pels on the other two channels are discarded. The next pel that will be supplied on each of these channels will be the first pel of the line.

Although it is convenient to think of two separate blocks (the horizontal alignment block and the horizontal upsampler filter) it is likely that the two will be implemented together. In order to explain the operation.

#### Upsampling Ratio

The upsampling ratio will be supplied to the filter as a two bit binary number. In order that the filter operates in a sensible manner, the upsampling ratio should be sampled, by the upsampler itself, once per field time. The circuitry supplying the ratio is then free to update the sampling ratio, in readiness for the next field, at any time during the current field.

The ratio should be sampled as the first pel of each field is actually accepted (rather than just after the last pel of the previous field). In this way, the very first field after reset (or after some pause in decoding) is upsampled with the correct ratio.

## Video Timing Generator

### Introduction

This section describes the video timing generator circuit (VTG 333) in accordance with the present invention. The VTG is primarily responsible for generating the various analogue video synchronizing signals, and also for maintaining knowledge of the display system's current raster position. This enables the VTG to provide controlling signals for the output multiplexer, which selects between active video, border and blanking sources for output. Both analogue and digital standards are supported, with two frame sizes (PAL and NTSC), and associated synchronizing behavior, selectable at setup. Border or cropping width will be specified in a token which will load a hardwired input to the VTG.

### Horizontal Timing

The horizontal timing parameters are illustrated in Figure 94. These are split into those that are fixed (for either PAL or NTSC) and those that are variable (i.e., the parameters associated with any borders or cropping that may be specified).

The interlaced nature of the video being displayed imposes a requirement for half-line based counting, so that various timing points are shown separately for each half of the line.

A line comprises an initial blanking period, the insertion of a SAV token, an active period, the insertion of an EAV token, and the trailing blanking period. During blanking lines, the active area will have blank values inserted rather than border and data.

A line sync pulse appears at the beginning of every line (HSYNC). On certain blanking lines, two sync pulses appear, one at the beginning and the other after the first half line. The width of these is dependent on which vertical region is active: equalization or serration (field sync).

During the initial horizontal blanking period, pels are discarded according to the cropping value (if the crop bit is set) - a fixed period of 120 cycles is allowed to discard the RHS cropped pels from the preceding line. The LHS pels for the current line are then discarded, and pels are stalled until the start of the active region. It is essential that there are no gaps in the data stream from which pels are being discarded, otherwise distortions will occur.



If the crop bit is not set, however, a border is constructed by inserting border value for a period of borderL, followed by data for picture width, and then border again until the end of the active region. Note that it is not necessary to calculate the borderR value.

The total horizontal border or crop width is specified in pels. The LHS border/crop value must be a multiple of 2 pels in order for the sampling to remain consistent. Consequently, it must be a multiple of 4 in terms of clock cycles. This can be achieved by masking out the least significant 2 bits from the original total border value in pels. For example, if the specified border is 91 pels, the left border will be 88 cycles long, and the picture width will be  $(720 - 91) \cdot 2$  cycles.

Streams of pels arriving at the output max are padded to give blocks of 32 pels. Considering this, together with the scaling factors to be supported, the maximum number of pels to be received for a line will be 832. This means that the maximum crop value will be 112 pels, giving 112 cycles of cropping at the LHS and the RHS.

#### Vertical Timing - PAL

The vertical timing parameters for PAL, in accordance with the present invention, are illustrated in Figure 95. Two fields are shown separately, as they have slightly different timing. Analogue parameters are indicated by the shaded regions, being identical for each field, and digital parameters are shown by the waveforms. For simplicity, the zero-border case is shown. If a non-zero vertical border is specified, border is inserted for a period of borderT, then data for picture height, then border again until the end of the active region (fixed). BorderT and picture height are calculated in a way analogous to borderL and picture width (in horizontal timing) respectively. Once again, the initial border (borderT) must be a multiple of 4, this time in terms of half lines because the top border must be a multiple of 2.

Note that MPEG codes 576 lines of video for PAL, whereas the analogue standard specifies only 525. This difference is accommodated by selecting data for output for 576 half-lines per field, but only asserting the analogue blanking signal for the requisite 575 lines.

#### Vertical Timing - NTSC

Next, NTSC vertical timing, in accordance with the present invention, is illustrated in Figure 96. It is similar in principle to the PAL timing, although slightly more complex. MPEG codes 480 lines of video for NTSC, whereas the analogue standard specifies 483. This

means that 3 lines of border must be inserted per frame to fill the gap (3 half lines per field). In addition, the judicial vertical blanking indicator, V, is specified in such a way that additional border lines are required to be inserted as padding before the active video lines. Non-zero vertical borders will be inserted in addition to those lines already indicated, as described in the previous section. Furthermore, note that vertical cropping is not allowed in either standard.

There is, at present, some uncertainty about the digital blank signal, V, since various reference sources give conflicting information. There are two main timing possibilities, illustrated by V and V', with the associated border select signals SB and SB', respectively.

#### VTG Structure

The video timing generator of the present invention comprises separate machines for the horizontal and vertical timing domains. The vertical machine provides control signals for the horizontal machine, which, in turn, provides the half-line increment signal for the vertical counters.

Inputs to the VTG are:

- clocks and reset
- PAL not NTSC
- horizontal border value with crop indicator
- vertical border value

Outputs are:

- horizontal, vertical and composite sync and blanking signals
- select signals for data, border, blanking
- a discard data bit for cropping
- insert SAV and EAV
- F and V values for construction of SAV and EAV
- a 2-bit YUV position counter for SAV/EAV insertion
- a firstline bit to indicate the start of a picture at startup

All of the outputs go to the output multiplexer block, including the sync signals which can then remain in synchronization with the data.

### Horizontal Machine

The horizontal machine is essentially a counter with hardware to detect the arrival of the various timing points as shown in Figure 94. The count goes from zero to half line length (which is different for PAL and NTSC) and is repeated for each half line. A hardwired comparator exists for each of the fixed timing points, these being activated according to the standard. In addition, there is a register for the border value (which is polled once per field), a subtractor to determine the picture width, and an auxiliary counter for counting down from the border value to zero. This procedure occurs in parallel with the main half-line counting. The datapath is 10 bits wide, and 15 hardwired comparators are required to implement both PAL and NTSC. The structure of this current embodiment is shown in Figure 97, together with approximate sizes. The datapath is estimated to be 360u x 330u.

In addition to the datapath, most of the control logic in the VTG of the present invention will be associated with the horizontal machine. This will probably amount to 100-200 gates.

Inputs to the horizontal machine are:

- clocks and reset
- horizontal border value and crop bit
- line, equalization or field sync indicators
- PAL not NTSC
- vertical blank
- insert vertical border

Outputs from the horizontal machine are:

- horizontal and composite blanks
- insert data
- insert border
- insert blank values
- discard input
- insert SAV or EAV, with YUV count
- hsync
- composite sync
- start of line
- half-line increment

### Vertical Machine

The vertical datapath has essentially the same structure as the horizontal datapath, but with 22 hardware comparators (8 for PAL, 14 for NTSC). The principal counter increments each half line, counting the half-lines through each half line, and counting the half-lines through each field, in turn. It is also 10 bits wide.

Moreover, it is advantageous for test purposes to multiplex the half-line pulse input with another, more frequent clock, so that the vertical machine can be run independently of the horizontal machine.

The estimated size is 360u x 420u.

Inputs to the vertical machine are:

- clocks and reset
- PAL not NTSC
- vertical border value
- half-kline increment

Outputs from the vertical machine are:

- select equalization, field or line sync
- vertical blank (analogue)
- vertical sync
- F, V and V' bits for SAV/EAV construction
- insert vertical border
- insert data
- insert blank value
- start of frame

### Hardwired Comparator Design

In the present invention, the hardwired comparator design is based on a string of series n-type transistors, either pre-charged or with pull-up, organized in a similar style to memory row decoders. Typically, these comparators will be about 8u high in the area estimates given.

### Output Multiplex

The output multiplex of the invention has the task of putting together the data for display. It combines data arriving from the earlier sections of the display pipeline with timing information obtained from the VTG.

The other input task of the output multiplex is to remove the two-wire interfacing. All the pipeline stages up to the output multiplex have a two-wire interface, indeed the data arriving at the input of the output multiplex will always arrive too early and will be stalled by taking accept low. However, there is no two-wire interface at the output of the device.

In order to achieve the above removal of the two-wire interfacing, the dynamics of the supply of data need to be sorted out so that the DRAM interface never stalls the data arriving at the output of the horizontal upsampler.

Basically, the output multiplex is making a decision on a field by field basis as to whether to output a field of data or not. At some point, close to the start of the first active line of the field, the output multiplex makes a decision. If there is valid data waiting at its input (i.e., *in\_accept* is low) then it will start to output the data. If, on the other hand, there is no valid data (for example, before the first picture has been decoded) then it will paint border color through the entire picture.

Actually, this procedure is slightly more complicated because the output multiplex must also ensure that the data is painted into the correct field. That is, there must be valid data waiting that belongs to the correct field before the display commences.

If at some point the data ceases to be valid, at a time that the output multiplex expects to have valid data available to paint into the display (which should never happen) then the output multiplex reverts to outputting border color which it continues to do for the remainder of the field.

#### Border Generation

Figure 98 shows the generation of border color to the left and right of the picture display in accordance with the present invention.

As shown, the VTG generates the border region at the left of the picture by asserting a signal that selects border color in the output multiplex. However, at the right hand side of the picture, the border color is generated by the output multiplex itself. It does this by

recognizing that it has "run out" of data and paints the remainder of the width of the picture in border color.

It must be understood that there are two possible interpretations of "run out" of data. One is that the output data from horizontal upsampler is not valid. However, this is not what is meant here. In this case, one runs out of data after the pel that is marked by the *last\_in\_line* signal as going the last one in the line has been included in the output stream. Figure 99 shows the equivalent action when clipping of the picture occurs.

As shown, the VTG signals to the output multiplex to clip pels to the left of the picture by asserting a signal to tell the output multiplex to discard input pels. Once this has occurred, the VTG will signal that the output multiplex should start to output the remaining pels. At the end of the active line (i.e., 720 pels later) the VTG de-asserts the signal and the output multiplex discards any remaining pels in the data on its input. Note that, in general, there will be a gap (in time) between the time when the VTG indicates that cropping should occur and the start of the active line. This significantly simplifies the design of the VTG. The output multiplex discards pels when the crop signal is asserted and then waits until the start of the active line period.

#### Output multiplex

The output multiplex controls the multiplexing of various sources of data together to form a CCIR 601 8 bit multiplexed data stream.

The timing (i.e., what is multiplexed in and when) is largely controlled by the VTG. The output multiplex is concerned with higher level issues. For example, at the start of decoding, when no pictures are available for display, the output multiplex will be painting border color throughout the entire image. Eventually, the first decoded picture will arrive at the output of the horizontal upsampler. Typically, this will not occur conveniently at the start of the field. The output multiplex asks once per field time "is there valid data ready for display?". If not, it waits for the next field to occur (and any valid data that happens to turn up in the meantime has to wait for the start of the next field).

The output multiplex also ensures that the correct field of data arriving from the SDRAM interface is painted into the correct field of the PAL or NTSC raster.

In addition, to dealing with the data, the output multiplex also selects the correct sync and blanking signals for outputting to the pins. This facilitates easy connection to a wide range

of composite encoders, DAC's, and the like. The registers for the output multiplex are as shown in Table 48. The bits for the output multiplex control are illustrated in Table 49.

There are four bytes of MPI registers associated with the output multiplex:

Register Name	Size/Dir.	Reset State	Description
border_cb	8	0xC0	Cb component of border color
border_y	8	0x80	Y component of border color
border_cr	8	0x40	Cr component of border color
outmux_ctrl	8	zero	

Table 48. Outmux registers

Register Name	Dir.	Reset State	Description
hs/cs	0	0	Controls whether horizontal sync or composite sync is present on the hcsync pin. 0 selects composite sync 1 selects horizontal sync
hcsync_ah	1	0	Controls the parity of the hcsync pin. 0 selects active low 1 selects active high

Register Name	Bit	Reset State	Description
vsync_ah	2	0	Controls the parity of the vsync pin. 0 selects active low 1 selects active high
cblank_ah	3	0	Controls the parity of the cblank pin. 0 selects active low 1 selects active high
blanking601	4	0	Controls and value of <sup>blanking</sup> data that is output during blanking. 0 selects the value zero 1 selects the value 0x10 (sixteen) For CCIR 601 data this pin must be set to 1.
enbl_sav_eav	5	0	Controls the generation of SAV and EAV control words in the output stream. 0 suppresses SAV and EAV, in which case, blanking values are output at the times when SAV and EAV would otherwise be generated. 1 enable SAV and EAV. Note that blanking601 should also be set to 1 to avoid the value zero appearing at the output except during SAV and EAV. For CCIR 601 data this pin must be set to 1.



Register Name	Bit	I State	Description
blank_screen	6	0	When set to 1, this bit causes border color to be painted over the entire screen, thereby blanking the screen. Note that decoding continues as normal, but the decoded pictures are rendered invisible.
vblank	7	—	This is a read-only bit (data written to this bit is ignored). It indicates vertical blanking.

Table 49. Bits from Outmux\_Ctrl

- a. Irrespective of the setting of this bit, chrominance data (both Cb and Cr) will be 0x80 (128 decimal) during blanking.

### Video Decoder Specifications and Features

In addition to the aforementioned detailed description, the following disclosure is also provided regarding a preferred embodiment of a video decoder suitable for practice of the invention.

- 
- |                               |                                  |
|-------------------------------|----------------------------------|
| • MPEG-2 MP @ ML              | • 2/3 and 1/1 pull down          |
| • Single 16 Mbit SDRAM        | • Video scaling                  |
| • High resolution MPEG-1      | • Power including SDRAM = 2.5 W  |
| • or Vision compatible        | • Self configuring               |
| • Automatic error concealment | • Small board area               |
| • Channel change support      | • QuietPad™ outputs              |
| • Time stamp management       | • On-chip video timing generator |
- 

The present invention includes a highly integrated, easy to use, MPEG-2 video decoder. It fully supports all the requirements of MPEG-2 Main Profile at Main Level.

The system of present invention is also self configuring (a single pin selects between PAL and NTSC operation) and, in many applications, can start-up and maintain video decoding with no external software support. Error concealment and recovery is fully automatic. More demanding applications may utilize the advanced features controlled by software running on an external microprocessor.

The present invention stores its own microcode in an on-chip ROM, thus avoiding the need to use an external ROM or download microcode before decoding can commence. See Figure 100.

The following more detailed description of the system of the present invention is set forth for purposes of organization, clarity and convenience of explanation under the headings listed below:

Signals .....	
Register map .....	
Power supplies .....	
Logic levels .....	
Clock signals .....	
Reset signals .....	
Coded data interface signals .....	
Supply data via the microprocessor interface .....	
Switching between input modes .....	
Rate of accepting coded data .....	
Coded data interface timing .....	
CDCLOCK .....	
Video output signals .....	
Video output control registers .....	
Borders, scaling and cropping .....	
Video output control registers .....	
Video signal timing .....	
MPI signals .....	
MPI electrical specifications .....	
Interrupts .....	
Page register .....	
SDRAM interface signals .....	
SDRAM configurations .....	
Connection of JTAG pins in non-JTAG systems .....	
Supported instructions .....	
Characteristics .....	
Level of Conformance to IEEE 1149.1 .....	
Start code detector registers .....	
Detection of start codes .....	
discard_all facility .....	
flag_picture_end facility .....	
start_code_search facility .....	
SCD example - channel change .....	
Parser registers .....	
Error codes .....	
Dealing with user data .....	
System organization .....	
Signals and registers .....	
Electrical specifications .....	
Coded data interface .....	
Video output interface .....	

Microprocessor interface .....  
Synchronous DRAM interface .....  
JTAG interface .....  
Start code detector .....  
Video parser .....  
Timestamp management .....  
Address generator configuration .....  
Mechanical information .....

This section includes a listing of all the signals (pins) used, in accordance with the present invention, and a listing of all the registers available through the microprocessor interface. (See Tables 50 and 51.)

### Signals

Signal Name	I/O	Pin Number	Description
CDCLOCK	I	137	Coded Data Interface. Used to supply coded data or Tokens to the system.
CD[7:0]	I	133, 132, 130, 129, 128, 127, 125, 124	
CDEXTN	I	134	
CDVALID	I	123	
CDACCEPT	O	122	
BMCDE	I	135	Micro Processor Interface (MPI)
ME[1:0]	I	99, 98	
MRAW	I	97	
MA[5:0]	I	107, 106, 104, 103, 102, 101	
MD[7:0]	O	119, 118, 117, 116, 114, 113, 112, 111	
IRQ	O	96	SDRAM Interface
DD[15:0]	I/O	36, 35, 33, 32, 30, 29, 27, 26, 21, 20, 18, 17, 15, 14, 12, 11	
DA[10:0]	O	152, 153, 143, 144, 146, 147, 149, 150, 159, 158, 156, 153	
BS	O		
DCKE	O	39	
DCLKOUT	O	38	
DCLKIN	I	23	
DWE	O	9	
DCAS	O	8	
DRAS	O	6	
DCS[1:0]	O	3, 2	
Y[7:0]	O	52, 53, 54, 55, 57, 58, 59, 60	Video output interface

Signal Name	I/O	Pin Number	Description
C[7:0]	O	42, 43, 44, 45, 47, 48, 49, 50	
HCSYNC	O	62	
VSYN	O	63	
YE	O	64	
CB/CR	O	65	
V16/8	I	67	
NTSC/PAL	I	68	
CBLANK	O	69	
VTGRESET	I	70	
TCK	I	74	JTAG port.
TDI	I	73	
TDO	O	72	
TMS	I	75	
TRST	I	79	
SYSCLK	I	139	
RESET	I	138	
TIMERRESET	I	82	
VCC	-	1, 7, 13, 19, 25, 31, 37, 142, 148, 154, 160	
VDD	-	46, 56, 76, 86, 95, 105, 115, 126, 136	
VDD	-	4, 10, 18, 22, 28, 34, 40, 41, 51, 61, 71, 80, 81, 91, 100, 110, 120, 121, 131, 140, 145, 151, 157	

Table 50. Signals

Signal Name	VO	Pin Number	Description
TPH0ISH	I	87	
TPH1SH	I	88	
TSTRSTCTRL	I	77	
TLOOP	I	78	Connect to GND or VDD during normal operation
PLLSELECT	I	83	If PLLSELECT = 0 the on-chip phase locked loops are disabled. Set PLLSELECT = 1 for normal operation.
PLLLOCK	O	84	
TDCLK	I	85	

Table 51. Test Signals

### Register Map

The register map of the present invention is divided into areas. The first 32 locations are required for the normal operation of the system. There is only five bits of address.

The next set of 32 locations are those in the address generation circuitry that are required to setup a non-default SDRAM memory map.

The remainder of the register map are registers that are only used for test and diagnostic purposes. These can be paged in instead of the address generator registers.

Table 52 illustrates the register map of the present invention.

Address (hex)	Interrupt Service	See
0x00 ... 0x03	Interrupt service	
0x04 ... 0x05	Input circuit	
0x06 ... 0x07	Start code detector	
0x08 ... 0x0e	Timestamp insertion	
0x0b ... 0x0f	(not used)	
0x10 ... 0x17	Parser	

Address (hex)	Interrupt Service	See
0x1b ... 0x1c	Output control	
0x1d	PLL control	
0x1e	DRAM PAD drive strength	
0x1f	page_select*	Table 3-4
0x20 ... 0x3f	paged register access	

Table 52. Overview of Register Map of Present Invention

\* In normal operation, page\_select should hold the value zero. In this case, locations 0x20 ... 0x3f will contain the address generation user registers.

Table 53 depicts the page select register.

page-select	Registers Selected	See
0	Addrgen user configuration registers	Table 3-5
1	Built in self test and IDCT test registers	Table 3-11 Table 3-12
2	IM_plus test registers and SCD test registers	Table 3-13 Table 3-14
3	Parser test registers	Table 3-15
4	Field/Frame test registers	Table 3-16
5	BOB test registers	Table 3-17
6	more BOB test registers	Table 3-17
7	Addrgen test registers	Table 3-18
8	DRAMIF test registers	Table 3-19

Table 53. Page Select Register

Table 54 illustrates the interrupt service area.



Address (hex)	Bit No.	Register Name	See Page
0x00	7	chip_event	
	6	end_search_event	
	5	unrecognized_start_event	
	4	flag_picture_end_event	
	3	parser_event	
	2		
	1		
	0		
0x01	7	chip_mask	
	6	end_search_mask	
	5	unrecognized_start_mask	
	4	flag_picture_end_mask	
	3	parser_mask	
	2		
	1		
	0		
0x02	7	idct_too_few_event	
	6	idct_too_many_event	
	5		
	4		
	3		
	2		
	1		
	0	watchdog_event	

Address (hex)	Bit No.	Register Name	See Page
0x03	7	idcl_too_few_mask	
	6	idcl_too_many_mask	
	5		
	4		
	3		
	2		
	1		
	0	watchdog_mask	

Table 54. Interrupt Service Area

Table 55 shows the input circuit registers of the present invention.

Address (hex)	Bit No.	Register Name	See Page
0x04	7	coded_busy	
	6	enable_mpi_input	
	5	coded_extn	
	4:0	(not used)	
0x05	7:0	coded_data	

Table 55. Input Circuit Registers

Table 56 shows the start code detector register of the present invention.

Address (hex)	Bit No.	Register Name	See Page
0x06	7	scdp_access	
	6	(not used)	
	5	discard_extension	
	4	discard_user	
	3	after_search_stop	
	2	flag_picture_end	

Address (hex)	Bit No.	Register Name	See Page
0x07	1	after_picture_slop	
	0	after_picture_discard	
	7:3	(not used)	
	2	discard_all	
	1:0	start_code_search	

Table 56. Start Code Detector Registers

In accordance with the present invention, Table 57 shows the timestamp insertion registers.

Address (hex)	Bit No.	Register Name	See Page
0x08	7:0	ts_high	
0x09	7:0	ts_low	
0x0a	7	ts_valid	
	6	ts_waiting	
	5:0	(not used)	

Table 57. Timestamp Insertion Registers

Likewise, Table 58 illustrates the video parser registers.

Address (hex)	Bit No.	Register Name	See Page
0x10	7:0	parser_ctrl0 (actually a reg file location - bits TBD)	
0x11	7:0	parser_ctrl1 (actually a reg file location - bits TBD)	
0x12	7:0	parser_error_code (actually const. field of MSM)	
0x13	7	parser_access	
	6:0	reg_keyhole_addr	
0x14	7:0	reg_keyhole_data	
0x15	7:0	(not used)	
0c16	7:0	user_keyhole_addr	

Address (hex)	Bit No.	Register Name	See Page
0x17	7:0	user_keyhole_data	

Table 58. Video Parser Registers

The output control registers are shown in Table 59.

Address (hex)	Bit No.	Register Name	See Page
0x18	7:0	border_cb	
0x19	7:0	border_y	
0x1a	7:0	border_cr	
0x1b	7	vblank	
	6	blank_screen	
	5	enbl_sav_eav	
	4	blanking601	
	3	cblank_ah	
	2	vsync_ah	
	1	hcsync_ah	
		hs_not_cs	
0x1c	7:2	(not used)	
	1:0	vertical upsample control	

Table 59. Output Control Registers

#### Test Registers

The complete register map is shown in Table 50 through Table 69.

Address (hex)	Bit No.	Register Name	See Page
P1+00		test_mode	
P1+01...P1+03		(not used)	
P1+04		misr_mask	
P1+05		(not used)	
P1+06		misr[1]	
P1+07		misr[0]	
P1+08		psrg_bit_select	
P1+09		psrg_constant	
P1+0a...P1+0c		(not used)	
P1+0d		psrg[2]	
P1+0e		psrg[1]	
P1+0f		psrg[0]	

Table 60. Built-in Self Test Registers

Address (hex)	Bit No.	Register Name	See Page
P1+10		idct_clkgen	
P1+11		(not used)	
P1+12		snp_idct[1]	
P1+13		snp_idct[0]	
P1+14...P1+17		not used	
P1+18		snp_1ram[7]	
P1+19		snp_1ram[6]	
P1+1a		snp_1ram[5]	
P1+1b		snp_1ram[4]	
P1+1c		snp_1ram[3]	
P1+1d		snp_1ram[2]	
P1+1e		snp_1ram[1]	

Address (hex)	Bit No.	Register Name	See Page
P1+1f		snp_ram[0]	

Table 61. IDCT Test Registers

Address (hex)	Bit No.	Register Name	See Page
P2+00		imp_clkgen	
P2+01		(not used)	
P2+02		snp_iquant[1]	
P2+03		snp_iquant[0]	
P2+04		(not used)	
P2+05		snp_imode[1]	
P2+06		snp_imode[1]	
P2+07		snp_imode[0]	
P2+08		snp_iquant_ram[3]	
P2+09		snp_iquant_ram[2]	
P2+0a		snp_iquant_ram[1]	
P2+0b		snp_iquant_ram[0]	
P2+0c		iquant_keyhole_data	
P2+0d		iquant_keyhole_addr	
P2+0e...P2+0f		(not used)	
P2+10		snp_izz_ram[3]	
P2+11		snp_izz_ram[2]	
P2+12		snp_izz_ram[1]	
P2+13		snp_izz_ram[0]	
P2+14		izz_keyhole_data	
P2+15		izz_keyhole_addr	
P2+16...P2+17		(not used)	

Table 62. IM\_plus Test Registers

Address (hex)	Bit No.	Register Name	See Page
P2+18		sca_clkgen	
P2+19		(not used)	
P2+1a		snp_incrct[1]	
P2+1b		snp_incrct[0]	
P2+1c		snp_cdbin[1]	
P2+1d		snp_cdbin[0]	
P2+1e...P2+1f		(not used)	

Table 63. SCD Test Registers

Address (hex)	Bit no.	Register name	See page
P3+00		parser_clkgen	
P3+01...P3+02		(not used)	
P3+03		snp_cdbout[4]	
P3+04		snp_cdbout[3]	
P3+05		snp_cdbout[2]	
P3+06		snp_cdbout[1]	
P3+07		snp_cdbout[0]	
P3+08		(not used)	
P3+09		snp_aluin[2]	
P3+0a		snp_aluin[1]	
P3+0b		snp_aluin[0]	
P3+0c...P3+0f		(not used)	
P3+10	7	msm_access	
	6:0	(not used)	
P3+11	7:3	(not used)	
	2	mssr_inir_status	
	1	mssr_ss_monitor	

Address (hex)	Bit no.	Register name	See page
	0	mssr_ss_select	
P3+12	7:4	(not used)	
	3:0	msm_pc	
P3+13	7:0		
P3+14	7:4	(not used)	
	3:0	msm_call_return	
P3+15	7:0		
P3+16	7:4	(not used)	
	3:0	msm_intr_return	
P3+17	7:0		
P3+18		snp_user_ram[7]	
P3+19		snp_user_ram[6]	
P3+1a		snp_user_ram[5]	
P3+1b		snp_user_ram[4]	
P3+1c		snp_user_ram[3]	
P3+1d		snp_user_ram[2]	
P3+1e		snp_user_ram[1]	
P3+1f		snp_user_ram[0]	

Table 64. Parser Test Registers

Address (hex)	Bit No.	Register Name	See Page
P4+00		ff_clkgen	
P4+01		(not used)	
P4+02		snp_fid_frm[1]	
P4+03		snp_fid_frm[0]	
P4+04		snp_padder_data[1]	
P4+05		snp_padder_data[0]	



Address (hex)	Bit No.	Register Name	See Page
P4+06		snp_padder_pf[1]	
P4+07		snp_padder_pf[0]	
P4+08		snp_pf_master[3] (snpse[3])	
P4+09		snp_pf_master[2] (snpse[2])	
P4+0a		snp_pf_master[1] (snpse[1])	
P4+0b		snp_pf_master[0] (snpse[0])	
P4+0c		snp_pf_slave[3] (snpse[7])	
P4+0d		snp_pf_slave[2] (snpse[6])	
P4+0e		snp_pf_slave[1] (snpse[5])	
P4+0f		snp_pf_slave[0] (snpse[4])	
P4+10		(not used)	
P4+11		snp_pf_pipe[2] (snpse[10])	
P4+12		snp_pf_pipe[1] (snpse[9])	
P4+13		snp_pf_pipe[0] (snpse[8])	
P4+14		ff_keyhole_data	
P4+15		ff_keyhole_addr	
P4+16		snp_dec_data[1]	
P4+17		snp_dec_data[0]	
P4+18		snp_ff_ram[7]	
P4+19		snp_ff_ram[6]	
P4+1a		snp_ff_ram[5]	

Address (hex)	Bit No.	Register Name	See Page
P4+1b		snp_fl_ram[4]	
P4+1c		snp_fl_ram[3]	
P4+1d		snp_fl_ram[2]	
P4+1e		snp_fl_ram[1]	
P4+1f		snp_fl_ram[0]	

Table 65. Field/Frame Test Registers

Address (hex)	Bit No.	Register Name	See Page
P5+00		bob_clkgen	
P5+01		(not used)	
P5+02		snp_vup_cb[1]	
P5+03		snp_vup_cb[0]	
P5+04		snp_vup_cr[1]	
P5+05		snp_vup_cr[0]	
P5+06		snp_hup_y[1]	
P5+07		snp_hup_y[0]	
P5+08		snp_hup_cb[1]	
P5+09		snp_hup_cb[0]	
P5+0a		snp_hup_cr[1]	
P5+0b		snp_hup_cr[0]	
P5+0c		(not used)	
P5+0d		snp_outmux[2]	
P5+0e		snp_outmux[1]	
P5+0f		snp_outmux[0]	
P5+10		(not used)	
P5+11		snp_vtq[2]	
P5+12		snp_vtq[1]	

Address (hex)	Bit No.	Register Name	See Page
P5+13		snp_vig[0]	
P5+14		snp_outiface[1]	
P5+15		snp_outiface[0]	
P5+16...P5+1f		(not used)	
P6+00...P6+07		snp_vupram_cb1[7:0] (bobupram)	
P6+08...P6+09		snp_vupram_cb0[7:0]	
P6+10...P6+17		snp_vupram_cr1[7:0]	
P6+18...P6+1f		snp_vupram_cr0[7:0]	

Table 66. BOB Test Registers

Address (hex)	Bit No.	Register Name	See Page
P7+0		addrngen_clkgen	
P7+1			
		sncopers	

Table 67. Addrngen Test Registers

Address (hex)	Bit no.	Register Name	See Page
P8+0		dram_clkgen	

Table 68. DRAMIF Test Registers

## Summary of Test Register Locations

Address (hex)	Data Bits	Register Name	Location
P2+1a...P2+1b	10	snp_increl[1:0]	The input of the chip (before the input circuit)
P2+1c...P2+1d	10	snp_cdbin[1:0]	Input of cdbin
P3+03...P3+07	33	snp_cdbout[4:0]	Input of cdbout
P3+09...P3+0b	19	snp_aluin[2:0]	Input of the ALU in the MSM

Address (hex)	Data Bits	Register Name	Location
P2+05...P2+07	19	snp_imodel[2:0]	Input of the inverse modeler
P2+02...P2+03	13	snp_quant[1:0]	Input of the inverse quantizer
P1+12...P1+13	13	snp_idct[1:0]	Input of the IDCT
P4+02...P4+03	10	snp_flg_frm[1:0]	Input of field-frame
P4+04...P4+05	10	snp_padder_data[1:0]	Transform data input of padder
P4+06...P4+07	8	snp_padder_pf[1:0]	Pred. filter data input of padder
P4+08...P4+0b	23	snp_padder_master[3:0]	Master input of predflt
P4+0c...P4+0f	23	snp_padder_slave[3:0]	Slave input of predflt
P4+11...P4+13		snp_pf_pipe[2:0]	Half way through predflt
P4+16...P4+17	8	snp_dec_data[1:0]	Output of prediction adder
P5+02...P5+03	10	snp_vup_cb[1:0]	Input of chroma upsample Cb
P5+04...P5+05		snp_vup_cr[1:0]	Input of chroma upsample Cr
P5+06...P5+07	12	snp_hup_y[1:0]	Input of horizontal upsampler y
P5+08...P5+09	10	snp_hup_cb[1:0]	Input of horizontal upsampler Cb
P5+0a...P5+0b	10	snp_hup_cr[1:0]	Input of horizontal upsampler Cr
P5+0d...P5+0f	10 + strobes from vlg	snp_outmux[2:0]	Input of outmux
P5+11...P5+13		snp_vlg[2:0]	All control inputs for VTG
P5+14...P5+15	13	snp_outface[1:0]	Just before 8 to 15 converter and retiming for the pins

Table 68. Snooper Registers

## Power Supplies

The present invention essentially operates from a single 5V supply. However, in order to enable simple connection to synchronous DRAM, a 3.3V supply is also provided.

Symbol	Parameter	Min.	Max.	Units
VDD	Nominal 5 V supply voltage relative to GND	-0.5	6.5	V
VCC	Nominal 3.3 V Supply voltage relative to GND	-0.5	6.5	V
V <sub>IN</sub>	Input voltage on any pin except SDRAM interface pins	GND - 0.5	VDD + 0.5	V
V <sub>PROGRAM</sub>	Input voltage on any SDRAM interface pin.*	GND - 0.5	VCC + 0.5	
T <sub>A</sub>	Operating temperature	-40	+85	°C
T <sub>S</sub>	Storage temperature	-55	+150	°C

Table 70. Suggested Specification Ratings \*

\* D[15:0], DA[11:0], DCKE, DCLKOUT, DCLKIN, DWE, DCAS, DRAS, DCS[1:0] and TDCLK

\* Stresses greater than those listed here may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these, or any other conditions above those indicated in the operational sections of this specification, is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

Symbol	Parameter	Min.	Max.	Units
VDD	Nominal 5 V supply voltage relative to GND	4.75	5.25	V
VCC	Nominal 3.3 V Supply voltage relative to GND	3.00	3.60	V
GND	Ground	0	0	V
T <sub>A</sub>	Operating temperature	0	70	°C*
I <sub>DD</sub>	RMS power supply current			mA

Table 71. DC Operating Conditions

### Logic Levels

Three different signal interface types are implemented in accordance with the present invention. Standard (5 V) TTL levels are employed by the microprocessor interface. In addition, 5 V CMOS levels are used by the coded data interface and the video output interface. 3 V LVTTTL levels are also employed by the SDRAM interface.

#### TTL (5 V) Levels

Symbol	Parameter	Min.	Max.	Units
V	Input logic '1' voltage	2.0	VDD + 0.5	V*
V <sub>IL</sub>	Input logic '0' voltage	GND - 0.5	0.8	V
V <sub>OL</sub>	Output logic '0' voltage		0.4	V
V <sub>OLOC</sub>	Open collector output logic '0' voltage		0.4	V*
V <sub>OH</sub>	Output logic '1' voltage	2.4		V
I <sub>O</sub>	Output current	± 100		μA*
I <sub>OC</sub>	Open collector output current	4.0	8.0	μA
I <sub>OL</sub>	Output off state leakage current		± 20	μA
I <sub>IL</sub>	Input leakage current		± 10	μA
C <sub>IN</sub>	Input capacitance		5	pF
C <sub>OUT</sub>	Output/I/O capacitance		5	pF

Table 72. TTL (5 V) DC Characteristics

- \* AC input parameters are measured at a 1.4 V measurement level
- \*  $I_O \leq I_{OCmax}$
- \* This is the steady state drive capability of the interface. Transient currents may be much greater.
- \* When asserted the open collector  $\overline{IRQ}$  output pulls down with an impedance of 100 Ω or less.

#### CMOS (5 V) Levels

For CMOS inputs  $V_{ILmax}$  is approximately 70% of  $V_{DD}$  and  $V_{OHmin}$  is approximately 30% of  $V_{DD}$ . The values shown in Table 73 are those for  $V_{IL}$  and  $V_{OH}$  at their respective extreme limits of operation.

Symbol	Parameter	Min.	Max.	Units
$V_{DHmin}$	Input logic '1' voltage	3.68	$V_{DD} + 0.5$	V
$V_{LHmin}$	Input logic '0' voltage	$GND - 0.5$	1.43	V
$V_{DHmax}$	Output logic '1' voltage	$V_{DD} - 0.1$		V <sup>a</sup>
		$V_{DD} - 0.4$		V <sup>a</sup>
$V_{OLmax}$	Output logic '0' voltage		0.1	V <sup>a</sup>
			0.4	V <sup>a</sup>
$I_{INmax}$	Input leakage current		$\pm 10$	$\mu A$
$C_{INmax}$	Input capacitance		5	pF
$C_{OUTmax}$	Output/I/O capacitance		5	pF

Table 73. CMOS (5 V) DC Characteristics

- <sup>a</sup>  $I_{IN} \leq 1 \text{ mA}$   
<sup>b</sup>  $I_{OL} \leq 4 \text{ mA}$   
<sup>c</sup>  $I_{IL} \leq 1 \text{ mA}$   
<sup>d</sup>  $I_{OL} \leq 4 \text{ mA}$

## LVTTTL (3.3 V) Levels

Symbol	Parameter	Min.	Max.	Units
$V_{IHmin}$	Input logic '1' voltage		$V_{CC} + 0.5$	V <sup>a</sup>
$V_{ILmax}$	Input logic '0' voltage	$GND - 0.5$	0.8	V
$V_{OLmin}$	Output logic '0' voltage			V
$V_{OHmin}$	Output logic '1' voltage			V
$I_{Omax}$	Output current	$\pm 100$		$\mu A^b$
$I_{OZmax}$	Output off state leakage current		$\pm 20$	$\mu A$
$I_{INmax}$	Input leakage current		$\pm 10$	$\mu A$
$C_{INmax}$	Input capacitance		5	pF
$C_{OUTmax}$	Output/I/O capacitance		5	pF

Table 74. LVTTTL (3.3 V) DC Characteristics

- <sup>a</sup> AC input parameters are measured at a V measurement level

- This is the steady state drive capability of the interface. Transient currents may be much greater.

### Clock Signals

The present invention uses one clock (SYSCLOCK) for almost all on-chip functions. Since this clock is used by the video output circuitry, it is assumed that a 27 MHz clock will be used so that the VTG (Video Timing Generator) will produce pictures at the correct rate.

A second clock (CDCLOCK) may be used to clock coded data into the present invention. This clock may be synchronous to SYSCLOCK and this allows data to be transferred into the system from circuitry that is not operating on the 27 MHz clock (perhaps a clock derived from a disk or network interface circuit).

Internally, the invention derives high speed clocks for driving the SDRAM interface using a phase locked loop (PLL). This clock is output of the SDRAM as DCLKOUT. An on-chip PLL is also used to derive an even mark-space ratio. The requirements for the SYSCLOCK are shown in Figure 101.

Num.	Characteristic	27 MHz		Unit	Note
		Min.	Max.		
1	Clock period	37		ns	
2	Clock high period	10		ns	
3	Clock low period	10		ns	

Table 75. Input Clock Requirements

- Note that the tolerance and stability of the clock must be adequate to comply with the line frequency of the appropriate video standard.

### Reset Signals

The present invention uses three reset signals:

- 1) RESET
- 2) VTGRESET
- 3) TIMERESET

RESET is the main chip reset signal. All circuitry is reset and adopts the reset state indicated in the various tables as described herein. RESET must be asserted (LOW) for at least four clock cycles after the power and clocks are stable to ensure a correct reset.



VTGRESET is used to reset the video timing generator of the present invention without affecting other aspects of the present invention.

TIMERESET is used by the timestamp handling circuitry in accordance with the present invention.

## Introduction

The coded data interface, in accordance with the present invention, provides a dedicated set of pins that may be used to supply the coded video data to system. Alternatively, coded data may be written via the microprocessor interface. This section discusses both of these methods.

If the dedicated pins are used, coded data may be supplied either as a simple stream of bytes or as "Tokens." The Tokens allow other types of information to be supplied in addition to the coded data. For example, time stamp information may be transferred using this mechanism.

If the microprocessor interface is used for coded data, then Tokens are always used. Furthermore, this is quite simple. Once a "Token Head" has been written to declare that subsequent data is coded data (requiring just two registers to be written) coded data may, thereafter, be simply written into a register.

## Coded Data Interface Signals

Table 76 defines the coded data interface signals used in the present invention.

Signal Name	Type	Description
CD[7:0]	I	Coded data is supplied to the present invention one byte at a time. Data is sampled at the rising edge of CDCLOCK. Data is assumed to be byte-aligned.
CDEXTN	I	When the coded data interface is used to transfer Tokens, this signal is the extension bit. This signal is sampled at the same time as CD[7:0].
CDVALID	I	CDVALID is sampled at the same time as CD[7:0]. When it is HIGH, the data is valid and is used as coded data. When it is LOW, the data is not valid and is ignored by the system.
CDACCEPT	O	CDACCEPT indicates the readiness of the system to accept data. When it is HIGH, at the rising edge of CDCLOCK data will be latched as expected. When it is LOW, the system cannot accept the data (presumably because its internal buffers are full) and, therefore, the data should be presented again.
BMODE	I	When this signal is HIGH, data is interpreted as a simple stream of coded data bytes (and CDEXTN is ignored). When it is low data is interpreted as Tokens. This signal is sampled at the same time as CD[7:0].

Signal Name	Type	Description
CDCLOCK	1	<p>This clock is used to control the transfer of data into the system. CD[7:0], CDEXTN, BMODE and CDVALID are sampled at the rising edge of CDCLOCK and external circuitry should sample CDACCEPT at the same time.</p> <p>Note that in the default (reset) condition, CDCLOCK and SYSCLOCK must be connected to the same signal.</p>

Table 76. Coded Data Interface Signals

CDVALID and CDACCEPT are used to control the transfer of data in accordance with the present invention. This type of protocol is referred to as a "two-wire" interface. Both signals must be high at the rising edge of CDCLOCK in order for a data transfer to occur. Figure 102 shows the relationship between the data (CD[7:0], CDEXTN and BMODE) and CDVALID and CDACCEPT.

Note: If data is to be supplied via the coded data interface pins, the microprocessor interface register "enable\_mpi\_input" must be zero (this is its reset state).

#### Byte Mode

In the present invention, if BMODE is sampled HIGH at the rising edge of CDCLOCK (and CDVALID and CDACCEPT are both high), then the data is treated as simple coded data. In fact, the data is immediately built into a DATA. In this case, CDEXTN is ignored.

#### Token Mode

If BMODE is sampled LOW, at the rising edge of CDCLOCK (and CDVALID and CDACCEPT are both high), then the data is treated as Tokens.

Tokens are used extensively in accordance with the present invention, to control the flow of data and control signals throughout the system. Theoretically, it is possible to supply any Token at the coded data input.

All Tokens, in accordance with the present invention, consist of a series of bytes (CD[7:0]), each of which has associated with it an extension bit (CDEXTN). The first byte of the Token indicates the type of information carried by the Token. The last byte of the Token is indicated by the extension bit being LOW.

For example, coded data is supplied using the DATA Token. This is illustrated in Figure 103. As shown, the first byte is 0x04 (indicating that this is a DATA Token). This information is followed by bytes of coded data that extend until CDEXTN is sampled LOW. The next data that is sampled will be interpreted as the first byte of a new Token (assuming that BMODE is still LOW).

Another Token that is particularly useful is the FLUSH Token. This Token acts like a "reset" and it may be used after the end of one video stream in order to ready the system for the next video stream. The FLUSH Token is illustrated in Figure 104.

### Supply Data via the Microprocessor Interface

In the present invention, tokens can be supplied to the system via the microprocessor interface (MPI) by accessing the coded data input registers. Table 77 defines the coded data input registers.

Addr. (Hex)	Bit No.	Dir/Reset	Register Name	Description
D4	7	RO/1	coded_busy	The state of this registers indicates if the system is able to accept Tokens written into coded_data[7:0].  The value 1 indicates that the interface is busy and unable to accept data. Behavior is undefined if the user tries to write to coded_data when coded_busy = 1.
	6	RW/0	enable_mpi_input	Controls whether coded data input to the system is via the coded data port (0) or via the MPI (1).
	5	RW/x	coded_extn	The extension bit of the token data written into coded_data.
	4:0	(not used)		
D5	7:0	RW/x	coded_data	Token data is written into this location.

Table 77. Coded Data Input Registers

### Writing Tokens via the MPI

The coded data registers are grouped into two bytes within the memory map to allow for efficient data transfer. The 8 data bits, coded\_data[7:0], are in one location and the control registers, coded\_busy, enable\_mpi\_input and coded\_extn are in a second location. (See Table 56.)

When configured for Token input via the MPI, the current Token is extended with the current value of `coded_extn` each time a value is written into `coded_data[7:0]`. Software is responsible for setting `coded_extn` to 0 before the last word of any Token is written to `coded_data[7:0]`.

For example, a DATA Token is started by writing 1 into `coded_extn` and then 0x04 into `coded_data[7:0]`. The start of this new DATA Token then passes into the system for processing.

Each time a new 8 bit value is written to `coded_data[7:0]`, the current Token is extended. `coded_extn` need only be accessed again when terminating the current Token (for example, to introduce another Token). The last word of the current Token is indicated by writing 0 to `coded_extn` followed by writing the last word of the current Token into `coded_data[7:0]`.

Moreover, each time before writing to `coded_data[7:0]`, `coded_busy` should be inspected to see if the interface is ready to accept more data.

### Switching between Input Modes

Provided suitable precautions are observed, it is practical to dynamically change the data input mode. In general, the transfer of a Token via any one route should be completed before switching modes. These switching modes are shown in Table 76.

Previous Mode	Next Mode	Behavior
Byte	Token	The on-chip circuitry will use the last byte supplied in byte mode as the last byte of the DATA Token that it was constructing (i.e., the extension bit will be set to 0). Before accepting the next Token.
	MPI input	
Token	Byte	The off-circuitry supplying the Token in Token mode is responsible for completing the Token (i.e., with the extn bit of the last byte of information set to 0). Before selecting byte mode.
	MPI input	Access to input via the MPI will not be granted (i.e., coded_busy will remain set to 1) until the off-chip circuitry supplying the Token in Token mode has completed the Token (i.e., with the extension bit of the last byte of information set to 0).
MPI input	Byte	The control software must have completed the Token (i.e., with the extension bit of the last byte of information set to 0) before enable_mpl_input is set to 0.
	MPI input	

Table 78. Switching Data Input Modes

The first byte supplied in byte mode causes a DATA Token header to be generated on-chip. Any further bytes transferred in byte mode are appended to this DATA Token until the input mode changes. The MPI register bit coded\_busy and the signal coded\_accept indicated on which interface the system is willing to accept data. Correct observation of these signals should ensure that no data is lost.

### Rate of Accepting Coded Data

The input circuit of the present invention passes Tokens to the start coded detector. This analyses data in the DATA Tokens and its normal rate of processing is one byte per clock (of CDCLOCK). However, extra processing cycles are occasionally required. For example, when a start code is encountered in the coded data. When this occurs, CDACCEPT will go low to indicate that data cannot be accepted.

It follows that CDCLOCK must have a higher clock frequency than the rate at which bytes of data are to be supplied to the system. In many applications, it will be appropriate to use the same clock (typically 27 MHz) for both SYSCLOCK and CDCLOCK. One example is shown in figure 105.

## Coded Data Interface Timing

Similarly, Table 79 shows the coded data interface timing for the present invention.

Num.	Characteristic	27 MHz		Unit	Note
		Min.	Max.		
1	CDCLOCK cycle time	37		ns	
2	CDCLOCK low time	17		ns	*
3	CDCLOCK high time	17		ns	
4	CDACCEPT drive time		23	ns	*
5	CDACCEPT hold time	2		ns	
6*	Input signal set-up time	5		ns	
7	Input signal hold time	0		ns	

Table 79. Coded Data Interface Timing

- \* These timings need not be observed in some circumstances.
- \* Maximum signal loading is 20 pF.

The coded data interface uses CMOS levels.

## CDCLOCK

The transfer of data across the coded data interface is controlled by CDCLOCK which may be synchronous to the main video decoder clock (SYSCLOCK). This facility may be useful in allowing the system decoder to operate on a different clock to the video clock.

However, CDCLOCK is also used internally in the present invention to clock circuitry such as the start code detector. Since CDCLOCK does not have the benefit of a Phase Locked Loop (PLL) to ensure even mark-space ratio, external circuitry must be used to ensure this or the timing parameters 2 and 3 shown in Figure 105.

In situations where CDCLOCK and SYSCLOCK do not need to be synchronous, the facility exists to drive the internal circuitry such as the start code detector from the PLL rather than CDCLOCK. This frees the external circuitry from the need to guarantee the even mark-spaced ratio.

Figure 105 shows the internal arrangement which allows the even mark-space ratio clock generated by the PLL to be routed to the start code detector in place of CDCLOCK.

If `un_named_register` is 0 (reset condition), the start code detector is clocked from the PLL. In this case, both CDCLOCK and SYSCLOCK must be connected to the same signal. The AC timing requirements for SYSCLOCK.

If `un_named_register` is 1, the start code detector is clocked using CDCLOCK. In this case, CDCLOCK may be synchronous to SYSCLOCK. CDCLOCK must obey the timings as specified in Figure 105.



## Introduction

The video output interface of the invention implements a digital output interface that complies to CCIR Recommendations 601 and 656. All of the synchronization and blanking information is included, in the form of special code words (SAV and EAV), in the same byte-wide stream of data as the video information.

In addition, separate sync and blanking pins are provided so that the system may be connected directly to a wide range of devices (such as video DACs or NTSC encoders). The timing of these signals is suitable for the generation of a video signal that complies with CCIR Recommendation 624.

The video data may be time-multiplexed on a single byte-wide bus. Alternatively, a sixteen bit output mode is provided, in which case, the luminance data is output on one byte wide bus while the two color difference signals are time multiplexed on a second byte wide bus.

## Video Output Signals

Table 80 provides the signals for the video output interface, in accordance with the present invention.

Name	Type	Description
Y[7:0]	O	Luminance output data
C[7:0]	O	Cr/Cb output data
HCSYNC	O	Horizontal or composite sync. The microprocessor register <code>hs_not_cs</code> controls which sync is present on this pin. The register <code>hesync_ah</code> controls the polarity of this signal.
VSNC	O	Vertical sync. The register <code>vsync_ah</code> controls the polarity of this signal.
CBANK	O	Composite blanking. The register <code>cblank_ah</code> controls the polarity of this signal.
YE	O	When sampled high at the rising edge of <code>SYSCLOCK</code> , the Y (and in 16 bit mode the Cr or Cb) data is valid.

CB/CR	0	In 16 bit mode, this signal indicates which color component (Cr or Cb) is present on the C[7:0] pins when YE is sampled high.  In 8 bit mode the signal indicates which color component (Cr or Cb) is present on the Y[7:0] pins when YE is sampled low.
V16/8	1	Used to select the 16 or 8 bit output modes. 16 bit mode is selected when V16/8 is HIGH. 8 bit mode is selected when it is LOW.
NTSC/PAL	1	Selects which of two standard rasters are to be produced. When NTSC/PAL is HIGH, a 525-line raster is produced. When it is low, a 625 line raster is produced.  Note that this pin also affects other aspects of the operation of the present invention.
VTGRESET	1	This signal may be asserted to reset the on-chip Video Timing Generator. This may be used to lock the video timing to some external constraint.

Table 80: Video Output Interface Signals

Figure 107 shows the output timing in 16 bit mode. Figure 108 shows the output timing in 8 bit mode.

### Video Output Control Registers

Video output control registers, in accordance with the present invention as shown in Table 81.

Addr (Hex)	Bit no.	dir/reset	Register name	Description
18	7:0	RW/ 0xC0	border_cb	Cb component of border color
19	7:0	RW/ 0x80	border_y	Y component of border color
1A	7:0	RW/ 0x40	border_cr	Cr component of border color

Addr (Hex)	Bit no.	dir/reset	Register name	Description
1B	7	RO/x	vblank	This is a read-only bit (data written to this bit is ignored). It indicates vertical blanking.
	6	RW/O	blank_screen	When set to 1, this bit causes border color to be painted over the entire screen, thereby blanking the screen. Note that decoding continues as normal, but the decoded pictures are rendered invisible.
	5	RW/O	enbl_sav_eav	Controls the generation of SAV and EAV control words in the output stream.  0 suppresses SAV and EAV, in which case, blanking values are output at the times when SAV and EAV would otherwise be generated.  1 enables SAV and EAV. Note that blanking601 should also be set to 1 to avoid the value zero appearing at the output, except during SAV and EAV.  For CCIR 601 data, this pin must be set to 1.
	4	RW/O	blanking601	Controls the value of luminance <sup>a</sup> data that is output during blanking.  0 selects the value zero.  1 selects the value 0x10 (sixteen).  For CCIR 601 data, this pin must be set to 1.
1B	3	RW/O	cblank_ah	Controls the polarity of the CBLANK pin.  0 selects active low  1 selects active high

Addr (Hex)	Bit no.	dir/reset	Register name	Description
	2	RWD	vsync_ah	Controls the polarity of the VSYNC pin. 0 selects active low 1 selects active high
	1	RWD	hcsync_ah	Controls the polarity of the HCSYNC pin. 0 selects active low 1 selects active high
	0	RWD	hs_not_cs	Controls whether horizontal sync or composite sync is present on the HCSYNC pin. 0 selects composite sync 1 selects horizontal sync
1C				(VUP sample mode)

Table 81: Video Output Control Registers

- Irrespective of the setting of this bit chrominance data (both Cb and Cr) will be 0x80 (128 decimal) during blanking.

### Borders, Scaling and Cropping

The present invention attempts to always produce a picture for display that is 720 pels by either 480 lines (525 line raster) or 576 lines (625 line raster). The invention automatically scales the decoded picture in order to attempt to fill this area.

Since only a limited number of scale factors are supported, it will not always be possible to fill this area precisely. If the resulting picture is too small, then a border will be painted around the decoded picture. This border will be such that the decoded picture is in the center of the screen.

Conversely, if the scaling produces a picture that is too big, then the picture is cropped to enable it to be displayed properly. The displayed region is the center of the decoded

picture. This cropping is limited so that not more than approximately 10% of the decoded picture is cropped. If more than this would be lost, then a smaller scaling factor is used.

The border color may be selected by writing to the registers `border_cb`, `border_y` and `border_cr`. After the device is reset, and before any pictures have been decoded, the entire screen will be filled with the border color. In addition, it is possible to paint border color over the entire screen by writing to `blank_screen`. This may be used to hide the video during, for instance, a channel change.

## Video Output Characteristics

### Characteristics

Figure 109 illustrates, in accordance with the present invention, the timing of the video output interface. Similarly, Table 82 illustrates the video output interface timing.

Num.	Characteristic	27 MHz		Unit	Note
		Min.	Max.		
8	Output drive time		23	ns	*
9	Output hold time	2		ns	
10	VTGRESET set-up time	5		ns	*
11	VTGRESET hold time	0		ns	

Table 82: Video output interface timing

- \* Maximum signal loading is 50 pF
- \* Failure to meet this timing parameter will simply lead to uncertainty in the precise clock cycle, on which the reset will occur. VTCRESET is provided with an on-chip synchronizer that will guard against metastability problems in the event that this timing parameter is not observed.

Table 83 defines video output mode signals. Figure 110 shows the video output mode signals.

Num.	Characteristic	27 MHz		Unit	Note
		Min.	Max.		
12	Setup before first clock after reset	5		ns	.

Table B3: Video Output Mode Signals

- \* Operation is undefined if NTSC/PAL or V16/8 change state after reset.

### Video Signal Timing

The video timing of the present invention is such that the resulting video output complies with the following CCIR recommendations:

- CCIR Recommendation 601
- CCIR Recommendation 656
- CCIR Recommendation 624

### Horizontal Timing

The horizontal timing is shown in Figure 111. The numbers are in SYSCLK cycles for the 525 line system (625 line system (625 line system in parentheses).

During equalization, the HSYNC signal is LOW for 62 cycles (66 cycles in the 625 line system).

During field synchronization, the HSYNC signal is LOW for 732 cycles (738 cycles in the 625 line system).

### Vertical Timing

The vertical timing is illustrated in Figure 112 for the 525 line (NTSC) system and Figure 113 for the 625 line (PAL) system. In these drawings the numbers down the left hand side provide the line number as per CCIR Rec. 656. The two columns at the right provide the "I" and "V" bits to be found in the SAV and EAV codes (see CCIR Rec. 601).

The smaller numbers in the center of the thick, solid, black lines provide the logical line numbers of the decoded MPEG picture. These are, therefore, numbered 0 to 479 for the 480 line used in the 525 line (NTSC) system and 90 to 575 for the 576 lines used in the 625 line (PAL) system.

Figure 114 shows the timing of the sync and blanking pips for the 525 line system and Figure 115 for the 625 line system. Note that only one of HSYNC or CSYNC may be output (see hs\_not\_cs) and that the polarity of each of these signals may be inverted (see cblank\_ah, etc.).

**VTG Reset State**

In the invention, the VTG resets to the start of line 4 for the 525 line (NTSC) system and to the start of line 1 for the 625 line (PAL) system.

## Introduction

A standard byte wide microprocessor interface (MPI) is used in accordance with the present invention. The MPI operates synchronously to the various decoder chip clocks.

### MPI Signals

Table B4 depicts the MPI interface signals.

Signal Name	Type	Description
$\overline{ME}[1:0]$	Input	Two active low chip enables. Both must be low to enable accesses via the MPI.
MRW	Input	HIGH indicates a read from a register on the system. LOW indicates a write to a register on the system.  This signal should be stable while the chip is enabled.
MA[5:0]	Input	Address specifies one of the locations in the chip's register map.  This signal should be stable while the chip is enabled.
MD[7:0]	Output	8 bit wide data I/O port. These pins are high impedance if either enable signal is HIGH.
IRQ	Output	An active low, open collector, interrupt request signal.

Table B4. MPI Interface Signals

### MPI Electrical Specifications

#### DC Characteristics

See 2.2.1, "TTL (5 V) levels."

Figures 123 and 124 illustrate the read and write timing of the MPI, respectively.



## AC Characteristics

Table 85 shows the Read Timing for the MPI.

Num.	Characteristic	Min.	Max.	Unit	Notes*
13	Enable low period	100		ns	
14	Enable high period	50		ns	
15	Address or $\overline{rw}$ set-up to chip enable	0		ns	
16	Address or $\overline{rw}$ hold from chip disable	0		ns	
17	Output turn-on time	20		ns	
18	Read data access time		70	ns	*
19	Read data hold time	5		ns	
20	Read data turn-off time		20		

Table 85. Microprocessor Interface Read Timing

- \* The choice, in this example, of  $\overline{ME}[0]$  to start the cycle and  $\overline{ME}[1]$  to end it is arbitrary. These signals are of equal status.
- \* The access time is specified for a maximum load of 50 pF on each of MD[7:0]. Larger loads may increase the access time.

Likewise, Table 86 shows the write timing for the MPI.

Num.	Characteristic	Min.	Max.	Unit	Notes
21	Write data set-up time	15		ns	*
22	Write data hold time	0		ns	

Table 86. Microprocessor Interface Write Timing

- \* The choice, in this example, of  $\overline{enable}[0]$  to start the cycle and  $\overline{enable}[1]$  to end it is arbitrary. These signals are of equal status.

## Interrupts

"event" is the term used to describe an on-chip condition that a user might want to observe. An event could indicate an error condition or it could be informative to user software.

There are two single bit registers associated with each interrupt or "event". These are the condition event register and the condition mask register.

### Condition Event Register

The condition event register is a one bit read/write register whose value is set to one by a condition occurring within the circuit. The register is set to one even if the condition only existed transiently. The register is then guaranteed to remain set to one until the user's software resets it or the entire chip is reset.

- The register is set to zero by writing the value one
- Writing zero to the register leaves the register unaltered.
- The register must be set to zero by user software before another occurrence of this condition can be observed.
- The register will be reset to zero upon reset.

### Condition Mask Register

The condition mask register is a one bit read/write register which enables the generation of an interrupt request if the corresponding condition event register(s) is (are) set. If the condition event is already set when 1 is written to the condition mask register, an interrupt request will be issued immediately.

- The value 1 enables interrupts.
- The register clears to zero upon reset.

Unless stated otherwise, a block will stop operation after generating an interrupt request and will re-start soon after either the condition event or the condition mask register are cleared.

### Event and Mask Bits

In the present invention, event bits and mask bits are always grouped into corresponding bit positions in consecutive bytes in the register map (see Table 55). This allows interrupt service software to use the value read from the mask registers as a mask for the value in the event registers to identify which event generated the interrupt.

### The Chip Event and Mask

The present invention has a single "global" event bit that summarizes the event activity on the chip. The chip event register presents the OR of all the on-chip events that have 1 in their mask bit.

A 1 in the chip mask bit allows the chip to generate interrupts. A 0 in the chip mask bit presents any on-chip events generating interrupt requests.

Writing 1 or 0 to the chip event has no effect. It will only clear when all the events (enabled by a 1 in their mask bit) have been cleared.

### The IRQ Signal

The  $\overline{\text{IRQ}}$  signal in the invention is asserted if both the chip event bit and the chip event mask are set. The  $\overline{\text{IRQ}}$  signal is an active low, "open collector" output which requires an off-chip pull-up resistor. When active the  $\overline{\text{IRQ}}$  output is pulled down by an impedance of 100  $\Omega$  or less. A pull-up resistor of approximately 4 k $\Omega$  should be suitable for most applications.

### Page Register

In order to reduce the number of register address signals required by the present invention, a page register is employed to enable more than 64 registers to be addressed. This page register is at location 0x1f. Register locations 0x00 to 0x1f are not affected by the contents of the page register and are always present in the register map. Registers in locations 0x20 to 0x3f depend on the page register.

There are no paged registers that are required for normal device operation. The paged registers are, finally, only used for test purposes.

In the invention, the page register is reset to the value zero. The user should ensure that no other value is written to this register.

## Introduction

## SDRAM Interface Signals

Table 87 illustrates the SDRAM Interface Signals.

Signal Name	Type	Description
DD[15:0]	I/O	Data pins
DA[10:0]	O	Address pins
BS	O	Bank select. Often this is labeled as A[11] on 16 Mbit SDRAM parts
CKE	I	Clock enable
DCLKOUT	O	SDRAM clock output.
DCLKIN	I	Connect to DCLKOUT
DWE	O	Write enable
DCAS	O	Column address
DRAS	O	Row address
DCS[1:0]	O	Chip select. DCS[0] selects the first "bank" of SDRAM. If a second "bank" is used (see SDRAM configurations 1 and 2) then DCS[1] is also used.

Table 87. SDRAM Interface Signals

## SDRAM Configurations

Table 88 illustrates SDRAM configurations.

Configuration	SDRAM Packages	Total DRAM	Organization
0	1	16 Mbit	16 Mbit, 1 M by 16 bits
1	2	20 Mbit	16 Mbit, 1 M by 16 bits 4 Mbit, 256 k by 16 bits
2	2	32 Mbit	16 Mbit, 1 M by 16 bits 16 Mbit, 1 M by 16 bits
3	2	32 Mbit	16 Mbit, 2 M by 8 bits

Configuration	SDRAM Packages	Total DRAM	Organization
			16 Mbit 2 M by 8 bits

Table 88. SDRAM Configurations

## Configuration Zero

See Figure 116 for the Configuration Zero SDRAM Connection.

Figure 117 illustrates the configuration for one SDRAM connection. Similarly, Figures 118 and 119 depict a configuration of two and three SDRAM connections, respectively.

## Introduction

The system, in accordance with the present invention, fully supports the Joint Test Action Group (JTAG) "Standard Test Access Port and Boundary Scan Architecture", now adopted by the IEEE as standard 1149.1.

All JTAG operations are performed via the Test Access Port (TAP), which consists of five pins. The TREST (Test Reset) pin resets the JTAG circuitry to ensure that the device doesn't power-up in test mode. The TCK (Test Clock) pin is used to clock serial test patterns into the TDI (Test Data Input) pin, and out of the TDO (Test Data Output) pin. Furthermore, the operational mode of the JTAG circuitry is set by clocking the appropriate sequence of bits into the TMS (Test Mode Select) pin.

The JTAG standard is extensible to provide for additional features at the discretion of the chip manufacturer. In accordance with the present invention, there are 9 user instructions, including three JTAG mandatory instructions. The extra instructions allow a degree of internal device testing to be performed, and provide additional external test flexibility. For example, all device outputs may be made to float by a simple JTAG sequence. See Table 89.

### Connection of JTAG Pins in Non-JTAG Systems

Signal	Direction	Description
TRST	Input	This pin has an internal pull-up, but must be taken low at power-up even if the JTAG features are not being used. This may be achieved by connecting TRST in common with the chip reset pin RESET.
TDI	Input	These pins have internal pull-ups, and may be left disconnected if the JTAG circuitry is not being used.
TMS		
TCK	Input	This pin does not have a pull-up, and should be tied to ground if the JTAG circuitry is not used.
TDO	Output	High impedance except during JTAG scan operations. If JTAG is not being used, this pin may be left disconnected.

Table 89 How to Connect JTAG Inputs

### Supported Instructions

This section describes the instructions supported in this implementation of the present invention. See Tables 90, 91, and 92.

Instruction	Description
EXTEST	This is the most basic instruction. It applies data from the boundary scan chain to the PCB, and captures the response. It has a pre-defined instruction code, which is all-0's in the instruction register.
SAMPLE/ PRELOAD	This instruction allows the boundary-scan chain to be parallel-loaded from the device's pins, and shifted, without the boundary-scan chain being switched in, i.e. transparently to system operation. By this means, a "snapshot" of the state of the device's pins may be taken (external clock control required to avoid metastability), or the boundary-scan chain may be pre-loaded before switching over into EXTEST mode.  The instruction code for SAMPLE/PRELOAD may be chosen by the manufacturer.
BYPASS	This instruction selects the 1-bit bypass register, to by-pass the boundary scan chain, and thus reduce the length of bit-stream required to access other devices on the PCB. The instruction code is pre-defined as all-1's.

Table 90. Mandatory Instructions

Instruction	Description
INTTEST	This does the reverse of EXTEST, i.e. applies data from the boundary-scan chain to the chip core, and captures the response. The instruction code may be chosen by us. It is up to the user to devise suitable tests to make use of this capability

Table 91. Optional Instructions That Are Supported

The following optional JTAG instructions are not supported:

- 1) IDCODE
- 2) RUNBIST

Instruction	Description
FLOATBS	This instruction pre-sets the Boundary-scan register to contain '1' in all open-drain cells, and '0' in all others. The system operation is not affected. Since a '0' in an output cell causes the output to float, this is a quick way of disabling all outputs (a common requirement for PCB testing). The outputs will not float until an instruction is loaded which switches in the Boundary-scan chain, e.g. EXTEST. (If FLOATBS were to switch in the boundary-scan chain itself, unknown data would be driven out of the pins until the UPDATE_DR state.)
INEXTEST	Does the combination of INTEST and EXTEST. Perhaps not very useful as we have individual versions anyway. It may allow some users to devise a faster combined PCB/chip test. Many JTAG devices use this combined mode rather than separate versions.
SETBYP	Selects the Bypass register between TDI & TDO, but switches the Boundary-scan chain in. This allows the PCB test to set up a constant pattern on one device's pins, but still access other device's pins without having to reload the first device. The name is consistent with the same function in Texas Instrument's "Scope" JTAG devices.
SHIFTBN	Like SAMPLE/PRELOAD, but without the SAMPLE operation. Allows the current Boundary-scan contents to be shifted some more, without being overwritten. T.I. have this instruction in their Scope devices, but variously call it READBN or RBRNM, neither of which is very intuitive.
SHIFTBT	Like SHIFTBN, except that the Boundary-scan chain is switched in. Potentially more useful than SHIFTBN, in that it could be used for optimizing PCB test patterns for small bits of logic externally connected between JTAG devices. E.g. for a 2-input gate near the far-end of the chain, several test patterns could be queued-up in the Boundary-scan chain, and applied in turn. EXTEST, in contrast, overwrites the Boundary-scan contents on each scan cycle.

Table 92. Additional Public Instructions

## Allocation of Instruction Codes

There are 14 defined instructions altogether. Hence there is a 4-bit long instruction register, with 2 unassigned instructions. Unassigned instructions are aliases of the BYPASS instruction, in accordance with IEEE1149.1.

The full list of instructions and their codes is shown in Table 93.



Code	Instruction	Register shifted	Signals capture	B/SCAN register	Class
0000	EXTEST	B/Scan	InputPads / 0's	switched in	MANDATORY
0001	SAMPLE/ PRELOAD	B/Scan	All Pads	transparent	MANDATORY
0010	INTEST	B/Scan	0's / OutputPads	switched in	RECOMMENDED
0011	FLOATBS	B/Scan	0's	transparent	PUBLIC
0100	SHIFTBT	B/Scan	No change	switched in	PUBLIC
0101	SHIFTBN	B/Scan	No Change	transparent	PUBLIC
0110	INEXTST	B/Scan	All Pads	switched in	PUBLIC
0111	unassigned	Bypass	0	transparent	RESERVED
1000	PRIVATE				
1001	PRIVATE				
1010	SPDATAT	ScanData	Internal sigs	switched in	PRIVATE
1011	SPDATAN	ScanData	Internal sigs	transparent	PRIVATE
1100	SETBYP	Bypass	0	switched in	PUBLIC
1101	unassigned	Bypass	0	transparent	RESERVED
1110	BYPASS	Bypass	0	transparent	PUBLIC
1111	BYPASS	Bypass	0	transparent	MANDATORY

Table 93. JTAG Instruction Codes

## Level of Conformance to IEEE 1149.1

## Rules

ALL rules are adhered to, although the following should be noted:

Rules	Description
3.1.1(b)	The TRST pin is provided.
3.5.1(b)	Guaranteed for all public instructions (see IEEE 1149.1 5.2.1(c)).

Rules	Description
5.2.1©	Guaranteed for all public instructions. For some private instructions, the TDO pin may be active during any of the states Capture-DR, Extn1-DR & Pause-DR
5.3.1(a)	Power on-reset is achieved by use of the TRST pin.
6.2.1(e,f)	A code for the BYPASS instruction is loaded in the Test-Logic-Reset state.
7.1.1(d)	Un-allocated instruction codes are equivalent to BYPASS.
7.2.1(c)	There is no device ID register.
7.6.1(b)	Single-step operation requires external control of the system clock.
7.9.1(…)	There is no RUNBIST facility.
7.11.1(…)	There is no IDCODE instruction.
7.12.1(…)	There is no USERCODE instruction.
8.1.1(b)	There is no device identification register.
8.2.1(c)	Guaranteed for all public instructions. The apparent length of the path from TDI to TDO may change under certain circumstances while private instruction codes are loaded.
8.3.1(d-i)	Guaranteed for all public instructions. Data may be loaded at times other than on the rising edge of TCK while private instructions codes are loaded.
10.4.1(e)	During INTEST, the system clock pin must be controlled externally.
10.6.1(c)	During INTEST, output pins are controlled by data shifted in via TDI.

Table S4. JTAG Rules

## Recommendations

Recommendations	Description
3.2.1(b)	TCK is a high-impedance CMOS input.
3.3.1(c)	TMS has a high impedance pull-up.
3.6.1(d)	(Applies to use of chip).
3.7.1(e)	(Applies to use of chip).

Recommendations	Description
6.1.1(e)	The SAMPLE/PRELOAD instruction code is loaded during Capture-IR.
7.2.1(f)	The INTEST instruction is supported.
7.7.1(g)	Zeros are loaded at system output pins during EXTEST.
7.7.2(h)	All system outputs may be set high-impedance.
7.8.1(f)	Zeros are loaded at system input pins during INTEST.
8.1.1(d,e)	Design-specific test data registers are not publicly accessible.

Table 95. Recommendations Met

Recommendation	Description
10.4.1(f)	During EXTEST, the signal driven into the on-chip logic from the system clock pin is that supplied externally.

Table 96. Recommendations Not Implemented

## Permissions

Permissions	Description
3.2.1(c)	Guaranteed for all public instructions.
6.1.1(f)	The instruction register is not used to capture design-specific information.
7.2.1(g)	Several additional public instructions are provided.
7.3.1(a)	Several private instruction codes are allocated.
7.3.1(c)	(Rule?) Such instructions codes are documented.
7.4.1(f)	Additional codes perform identically to BYPASS
10.1.1(i)	Each output pin has its own 3-state control.
10.3.1(h)	A parallel latch is provided
10.3.1(i,j)	During EXTEST, input pins are controlled by data shifted in via TDI.
10.6.1(d,e)	3-state cells are not forced inactive in the Test-Logic-Reset state.

Table 97. Permissions Met

### Introduction

The start code detector (SCD), in accordance with the present invention, has the task of detecting start codes in the coded data stream. It converts these to Tokens for further internal processing by the system.

In addition to this task there are a series of features that support, for example, channel change.

### Start Code Detector Registers

Table 98 illustrates the registers for the start code detector of the present invention.

Addr (Hex)	Bit no.	Dir/reset	Register Name	Description
06	7	RW/0	scdp_access	<p>This bit must be set to one before the values in register location 0x07 may be written to reliably. This causes the SCD to stop processing data so that there is never any contention between the microprocessor access and any attempt by the SCD to modify the registers itself.</p> <p>Once the value one has been written to scdp_access, the microprocessor must poll scdp_access and wait until it reads back 1.</p> <p>Once the required accesses have been made to location 0x07, the value 0 should be written to scdp_access to enable the SCD to continue processing data.</p>
6			(not used)	

Addr. (Hex)	Bit no.	Dir/reset	Register Name	Description
	5	RW/1	discard_extension	When discard_extension is 1, any extension data that is not recognized as MPEG-2 MP@ML is discarded at the start code detector. When it is 0, such extension data is passed through the coded data buffer to the parser.  With the standard microcode, there is no point in setting discard_extension to 0.
	4	RW/1	discard_user	When discard_user is 1, any user data is discarded at the start code detector. When it is 0, user data is passed through the coded data buffer to the parser.  Whilst facilities exist to handle small amounts of user data at the parser, care must be exercised if discard_user is set to 0. Note that the system cannot deal with arbitrary amounts of user data.
	3	RW/0	after_search_stop	Used in conjunction with the start_code_search facility.
07	2	RW/0	flag_picture_end	This is set to 1 to enable the flag_picture_end facility.
	1	RW/0	after_picture_stop	Used in conjunction with the flag_picture_end facility.
	0	RW/0	after_picture_discard	Used in conjunction with the flag_picture_end facility.
	7:3	--	(not used)	
	2	RW/0	discard_all	This is set to 1 to enable the discard_all facility.

Add. (Hex)	Bit no.	Dir/reset	Register Name	Description
	1:0	RW/0	start_code_search	A non-zero value in this register enables the start_code_search facility. See 8.5 on page 84.
00	7	--	(not associated with the start code detector)	
	6	RW/0	end_search_event	This bit is set whenever a start_code_search is satisfied. If end_search_mask is also set to 1 then an interrupt will be generated.*
	5	RW/0	unrecognized_start_event	This bit is set whenever an unrecognized start code is detected. If unrecognized_start_mask is also set to 1, then an interrupt will be generated.
	4	RW/0	flag_picture_end_event	This bit is set whenever the end of a picture is detected and flag_picture_end=1. If flag_picture_end_mask is also set to 1 then an interrupt will be generated. See 8.4 on page 62.
	3:0	--	(not associated with the start code detector)	
01	7	--	(not associated with the start code detector)	
	6	RW/0	end_search_mask	See end_search_event above.
	5	RW/0	unrecognized_start_mask	See unrecognized_start_event above.
	4	RW/0	flag_picture_end_mask	See flag_picture_end_event above.
	3:0	--	(not associated with the start code detector)	

Table 98: Start code detector registers

- \* event bits are not simple R/W register bits
- \* all interrupts are conditional on chip\_mask being set to 1

### Detection of Start Codes

The start code detector of the present invention will only detect start codes that are correctly byte aligned.

The present invention deals only with video start codes. Unrecognized start codes are detected and cause an `unrecognized_start_code` event. The unrecognized start codes are the system start codes (with values 0xb9 through 0xf) the reserved start codes (0xb0, 0xb1, and 0xb6) and the `sequence_error_code` (0xb4).

### discard\_all Facility

The `discard_all` facility may be used to discard all data that enters the system. It is possible to select the `discard_all` facility "manually" by setting the register `discard_all` to 1. However, it is necessary that `scdp_access` must first be set to 1 and then polled until it reads back 1. Generally, it is typical to enter this mode automatically as part of the `flag_picture_end` facility.

The present invention will continue to discard all data until either the value 0 is written to `discard_all` or a FLUSH Token is encountered. Note that FLUSH Token that the resets `discard_all` is deleted from the stream of tokens and does not affect the parser or any subsequent blocks of circuitry.

### flag\_picture\_end Facility

The `flag_picture_end` facility, in accordance with the present invention, is intended to allow a clean termination of decoding by waiting until the end of a picture before stopping the flow of data into the system. The parser, therefore, will see no incomplete pictures.

Figure 120 illustrates as a flow chart the `flag_picture_end` facility. As shown, it is possible to generate an interrupt (`flag_picture_end_event`) when the end of the picture is detected. This may cause the SCD to stop processing data until the interrupt is serviced. Alternatively, the SCD may be allowed to proceed.

If `after-picture_discard` is set to 1, then after the end of the picture is detected, all subsequent data will be discarded. This is most useful for discarding the trailing data from one channel that is "in flight" in the system demultiplexor prior to a channel change.

Note that the `start_code_search` facility in this embodiment takes priority over `flag_picture_end` facility. In this way, the data that is being discarded due to the `start_code_search` is not examined to determine whether the end of a picture has been reached.



### start\_code\_search Facility

In the invention, the SCD can be set to search for specific types of start codes. This may be used, for instance, after a channel change to search for a sequence start code before decoding commences.

start_code_search	Start codes that end the search
0	(none - normal operation)
1	picture_start_code, group_start_code and sequence_start_code
2	group_start_code and sequence_start_code
3	sequence_start_code

Table 99. start\_code\_search Modes

The search mode is entered by writing a non-zero value into start\_code\_search. The start code detector will then search for the appropriate start codes as indicated by Table 99. All data and Tokens are discarded while the search continues. When one of the appropriate start codes is encountered, the search ends. start\_code\_search is set to zero and an interrupt may optionally be generated.

Note also that a FLUSH Token will terminate the search as if one of the indicated start codes had been encountered. However, in the special case that the FLUSH Token is terminating the discard\_all function, the search is not terminated. Further, this allows a direct transition between the discard\_all and a previously selected search mode when the FLUSH Token is encountered.

Figure 121 illustrates as a flow chart the start\_code\_search facility, in accordance with the present invention.

### SCD Example - Channel Change

An example of the using the SCD facilities in the invention is shown in the following sequence of actions which effect a channel change operation.

- 1) The controlling microprocessor recognizes the need for a channel change (perhaps responding to a signal from a remote control unit). The microprocessor will use the flag\_picture\_end facility of the SCD by writing:
  - 1 in to flag\_picture\_end
  - 1 in to after\_picture\_discard
  - 1 in to flag\_picture\_end\_mask

- 2) When the start code detector detects the end of the current picture, it immediately starts to discard all subsequent data. The microprocessor is interrupted and determines that the cause of the interrupt was flag\_picture\_end\_event. The microprocessor first prepares the start code detector for the new channel by writing:
  - 3 (search for sequence\_start) into start\_code\_search,
  - 1 to flag\_picture\_end\_event (to clear the event)
- 3) Then the microprocessor retunes the tuner to select the new channel.
- 4) After the last data from the old channel is transferred into the system (and before the first data from the new channel) a FLUSH Token is inserted. (Alternatively, the value 0 is written to discard\_all.) The start code detector, therefore, stops discarding the data (from the old channel) and starts searching (the data from the new channel) for a sequence start code.
- 5) Once the sequence start code is detected, the start code detector ceases discarding data and resumes normal decoding.

## Introduction

The video parser, in accordance with the present invention, is responsible for decoding the video data stream. It is implemented as a microprogrammed processor.

In the normal course of events, there is little need to interact with the video parser and many simple applications may simply let it get on with its job of decoding video.

However, the video parser is able to notify the controlling microprocessor when it detects unusual or unexpected events, such as bitstream errors. In all cases, the microcode includes code to recover from (and conceal) errors so that it is safe to ignore bitstream errors. However, the knowledge that bitstream errors are occurring may be useful for diagnostic purposes.

Furthermore, some aspects of Timestamp management are dealt with in the parser's microcode processor. These are documented in Chapter 10.

## Parser Registers

The registers used by the parser as shown in Table 100

Address (Hex)	Bit no.	Dir/reset	Register Name	Description
10	7:1	RW	parser_ctrl	No function allocated
	0	RW	parser_continue	Used in certain situations to indicate to the parser whether it should continue with its current activity or return to normal decoding.
11	7:0	RW	parser_status	Used to indicate the status of the parser in certain conditions
12	7:0	RO	parser_error_code	This location contains an error code when the parser has interrupted and is waiting to be serviced. This indicates the reason for the interrupt.

Address (Hex)	Bit no.	Direction	Register Name	Description
13	7	RW:0	parser_access	The value 1 must be written to this register to enable access to the other parser registers. The controlling microprocessor must then poll this bit until it reads back the value 1 indicating that the parser has stopped processing data and can be accessed.  Note that as a special case, if the parser is stopped waiting for it interrupt to be serviced parser_error_code may be read without first writing 1 to parser_access
	6:0	RW	reg_keyhole_addr	This register is used to address the location in the parser's internal register file that may be written to or read from via reg_keyhole_data. Note that each access (read or write) to reg_keyhole_data increments reg_keyhole_addr by one.
14	7:0	RW	reg_keyhole_data	A read from this location actually reads data from the parser's register file at the location indicated by reg_keyhole_addr. Similarly a write to this location actually writes to the parser's register file at the location indicated by reg_keyhole_addr.
15	7:0		(not used)	
16	7:0	RW	user_keyhole_addr	This register is used to address the location in the user data RAM that may be written to or read from via user_keyhole_data. Note that each access (read or write) to user_keyhole_data increments user_keyhole_addr by one.
17	7:0	RW	user_keyhole_data	A read from this location actually reads data from the user data RAM at the location indicated by reg_keyhole_addr. Similarly a write to this location actually writes to the user data RAM at the location indicated by reg_keyhole_addr.
00	7:0	--	(not associated with the parser)	

Address (Hex)	Bit no.	Dir/reset	Register Name	Description
	3	RW <sup>a</sup> /0	parser_event	This bit is set whenever the parser detects an error condition. If parser_mask is also set to 1 then an interrupt will be generated <sup>b</sup>
	2:0	--	(not associated with the parser)	
01	7:4	--	(not associated with the parser)	
	6	RW <sup>a</sup> /0	parser_mask	See parser_event above
	3:0	--	(not associated with the parser)	

Table 100. Parser Registers

- <sup>a</sup> event bits are not simple RW register bits.
- <sup>b</sup> all interrupts are conditional on chip\_mask being set to 1.

### Error Codges

Whenever the parser detects an event condition, it sets parser\_event. If parser\_mask is set to 1 (indicating that the user system is interested in servicing parser events) the parser stops processing and (assuming that chip\_mask is set to 1) an interrupt is generated.

On responding to the interrupt the controlling microprocessor should read parser\_error\_code to determine the cause of the event. Table 101 provides the complete list of defined error codes in this regard.

After the controlling microprocessor has responded to the event in the appropriate manner it must allow the parser of the present invention to resume processing. This is done by clearing the event by writing the value 1 to parser\_event.

Code	Name	Description
	ERR_USER_DATA	Indicates that user data has been encountered and is present in the user data RAM.

Table 101. Parser Error Codes

### Dealing with User Data

Small amounts of user data may be read out from the parser. By default, all user data is discarded by the scan code detector. This is to protect the system from the inappropriate use of large amounts of user data which would be beyond its capabilities.

In order to allow user data to reach the parser the register `discard_user` must be set 0. Whenever user data is encountered in the bitstream the bytes of data are buffered up in an on-chip user data RAM. The RAM has space for 192 bytes of data to be buffered. When all of the bytes of user data have been read (or the RAM is full) the parser will generate an event (`ERR_USER_DATA`) which allows the controlling microprocessor to read the data from the RAM.

Before the user data RAM is read, the microprocessor must first obtain access to the parser's internal registers by setting `parser_access` to 1 and then polling this bit until it reads back 1. The number of bytes in the user data RAM is indicated by `parser_status`. The user-data RAM cannot be accessed directly. Instead, it is necessary to write the address that is to be read into `user_keyhole_addr` (usually zero) then data is read from `user_keyhole_data`. Since `user_keyhole_addr` is automatically incremented each time that a read is performed from `user_keyhole_data`, the appropriate number of bytes of user data can be read very quickly.

If there are less than 192 bytes of user data, then all of the data is dealt with by a single event. If there are more than 192 bytes, then `parser_status` will contain 192 bytes the first time that `ERR_USER_DATA` is generated. After the event has been cleared (by writing zero to `parser_access` and then 1 to `parser_event`) the microcode will interrogate `parser_continue` to determine what to do next.

If `parser_continue` is 1 the parser continues dealing with user data. The remaining bytes of user data (or the next 192 bytes) are parsed from the stream and the process repeats. However, if `parser_continue` is 0 then the parser discards the remaining user data and proceeds with normal video decoding. Note that even if `parser_continue` is zero, the first `ERR_USER_DATA` event will always be generated.

#### Limiting the Amount of User Data

If it is intended that user data should be used, it is important that this is limited in order that the real-time decoding of video data can be guaranteed in accordance with the present invention. It is very difficult to specify the acceptable limit on user data since it depends on many external constraints such as the interrupt response time of the controlling microprocessor and the time taken to read a byte of data from the system. As a guide, the amount of user data should be limited to the amount that can be guaranteed to be read from the system in about 50 $\mu$ s (including interrupt response time etc.).

#### User Data RAM

During the decoding of picture data, the user RAM is used by the microcode processor for other purposes (storage of concealment motion vectors, for instance). For this reason, it is not possible to leave data in the RAM and expect it to be preserved for later use.

### Introduction

The present invention includes circuitry to assist in the management of video time stamps. It is assumed that the external circuitry associated with the MPEG system stream parser has recovered a stable 27 MHz clock by using the clock references (Programmed Clock Reference or System Clock Reference as appropriate).

The circuitry, in accordance with the present invention, is, therefore, concerned with starting video decoding at the appropriate time to ensure synchronization with audio and, thereafter, monitoring video timestamps to ensure continued synchronization. In the absence of errors, no subsequent correction will be required.

It is desirable to avoid the need to transfer clock reference information into the video decoder. Hardware is divided into two areas, a circuit associated with the input stages of the system for loading video time stamps and a real-time counter that is associated with the video parser circuitry.

### System Organization

The present invention includes a counter that is incremented at regular intervals derived from the 27 MHz SYSCLK. The system for timestamp management depends (conceptually) on a second copy of this counter being maintained outside of the system. These two counters are initialized to the same value by being reset by the same signal. Thereafter, the two counters free-run.

The present invention performs its timestamp management with respect to its internal time counter denoted "videotime". To assure that the correct comparisons are made, the video timestamps are modified by the system decoder. It is not necessary to know the absolute time - simply the difference between the actual time that a picture is decoded and the nominal time it should have been decoded.

Equation 1 below denotes this by setting the difference between the video time counter and the modified time stamp equal to the difference between the actual "time" (derived from the clock references) and the timestamp. Equation 2 is merely a reorganization of the variables to derive the modified time stamp.

EQ 1:

$$\text{videotime} - \text{modifiedtimestamp} = \text{timestamp} - \text{time}$$

EQ 2:

$$\text{modifiedtimestamp} = \text{videotime} + (\text{timestamp} - \text{time})$$

Figure 122 shows one possible organization of the arithmetic to derive the modified time stamp. In reality, it is most likely that the actual additions (and the shift) will be performed on a processor rather than in dedicated hardware. There are, of course, many other ways to derive the same numeric value of the modified time stamp. For instance, rather than



having a copy of the videotime counter, it may be better to simply record the value of "time", when the RESET\_TIME pin of the invention was last asserted. From this information and the current value of "time" it is always possible to deduce the current contents of video time within the system.

It will be appreciated that any suitable rearrangement of arithmetic operations that yields a suitable value of the modified time stamp may be used.

As shown in Figure 122, the modified timestamps used by the invention use only sixteen bits. This is achieved in two ways. First, since the difference between time and the timestamp (used to derive the modified timestamp - see Equation 2) should always be small, the more significant bits can be discarded. Second, since the invention only controls the presentation of video to the nearest frame-time, the less significant bits are also not required and are discarded by shifting right by four bits.

Thus, the sixteen bits of time information maintained are able to deal with timing errors of up to about 11.5 seconds with an accuracy of about 180  $\mu$ s (about 1% of a field time).

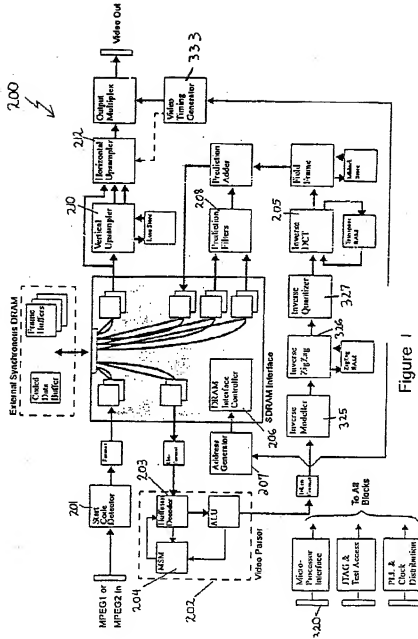


Figure 1

Figure 3

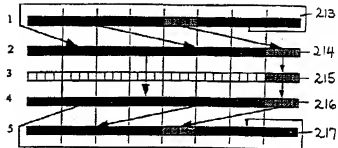


Figure 4

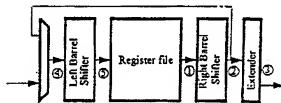


Figure 5

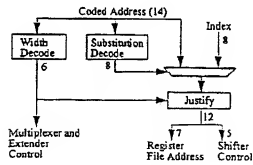


Figure 6

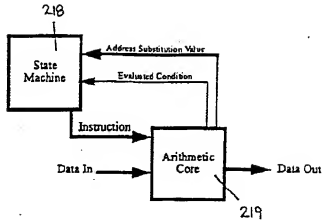


Figure 8

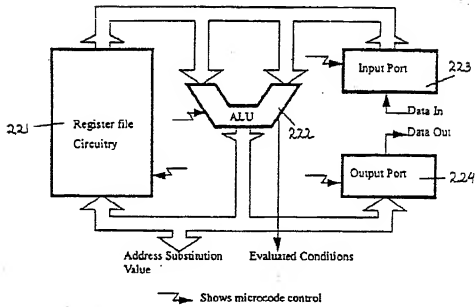
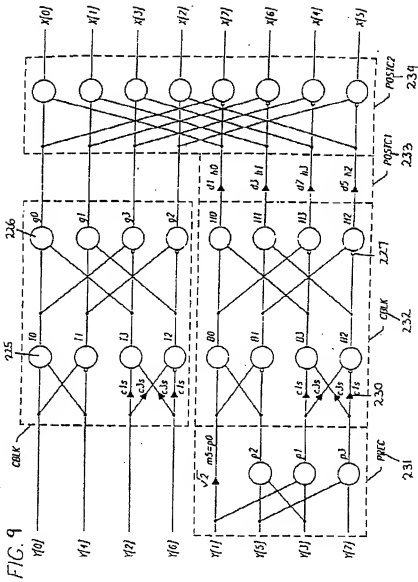


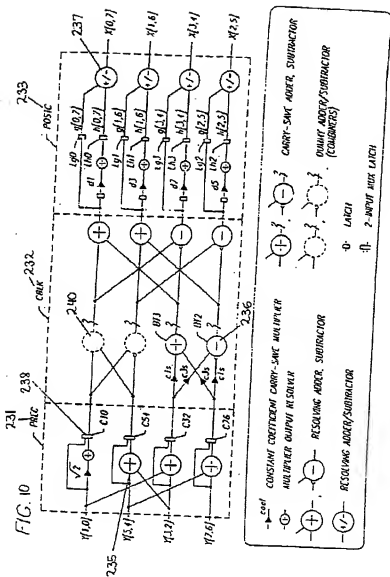
Figure 1

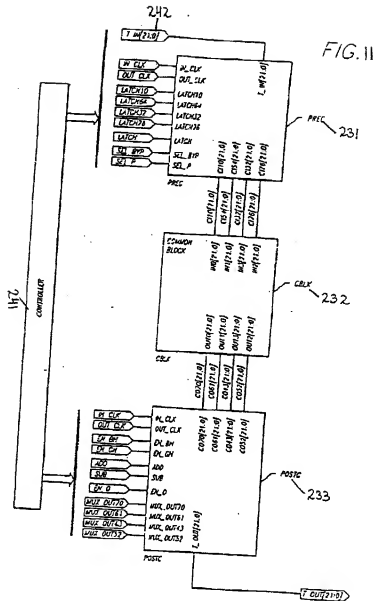
Fixed Width Word for Addressing			
Address Field		Substitution Indicator	
		Substitution Field	
	Termination marker	Continuation markers	
aa.....aa	yy.....yy	xx.....xx	ww.....ww

Figure 2

Fixed Width Word for Addressing					
Width Defining Field		Address Field		Substitution Indicator	
				Substitution Field	
Continuation markers	Termination marker			Termination marker	Continuation markers
uu.....uu	vv.....vv	aa.....aa	yy.....yy	xx.....xx	ww.....ww
000	1	1101	1	000	0
111	0	1101	0	111	1











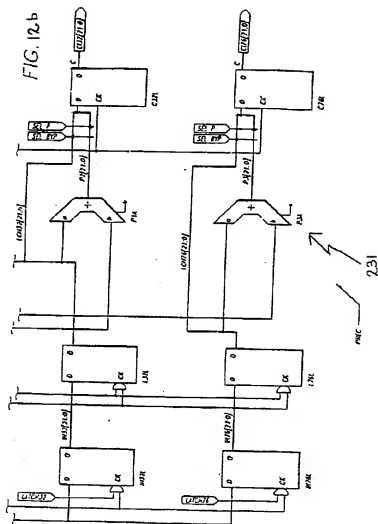


FIG. 13a

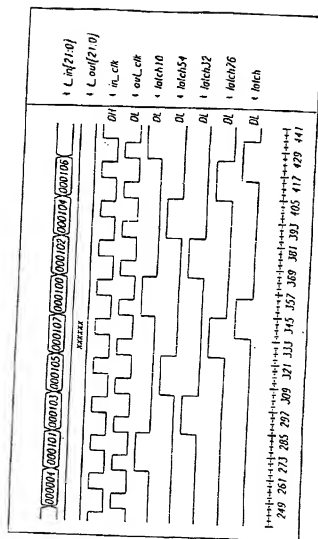




FIG. 13c

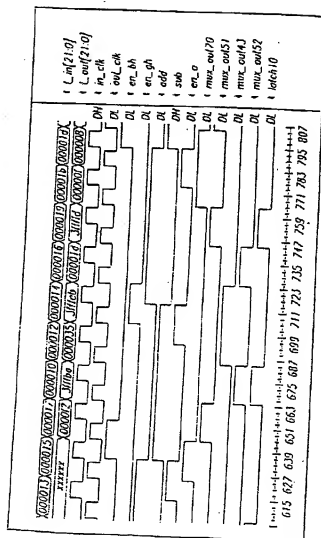
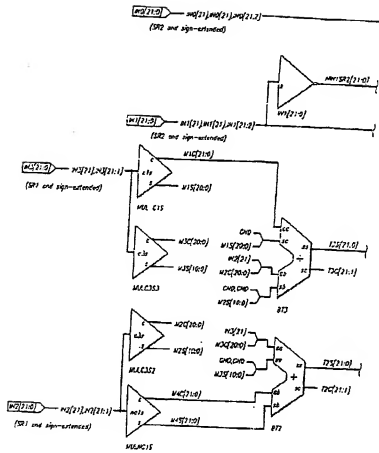
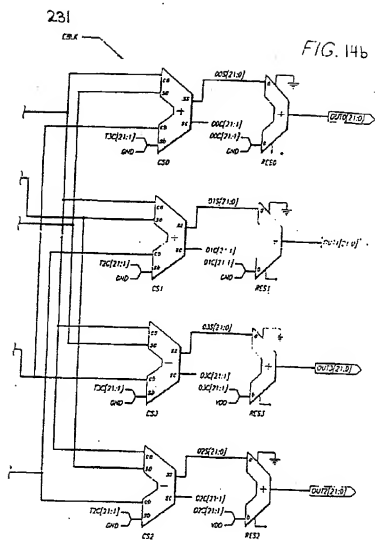
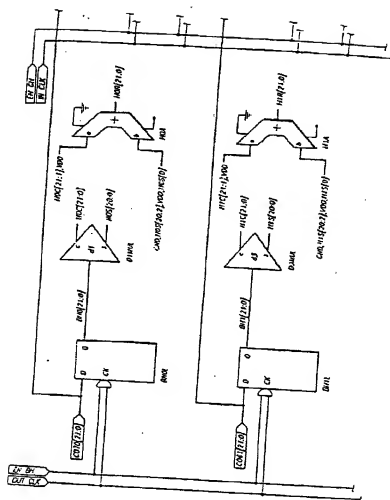


FIG. 14a





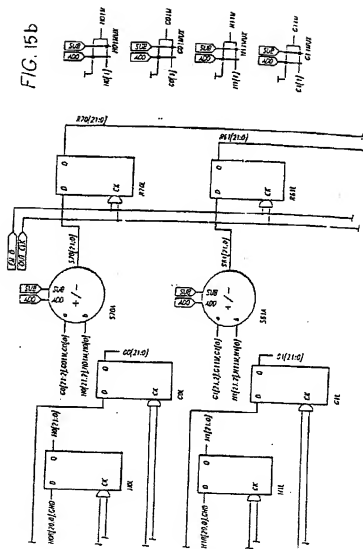


233

FIG 15a



FIG. 15b



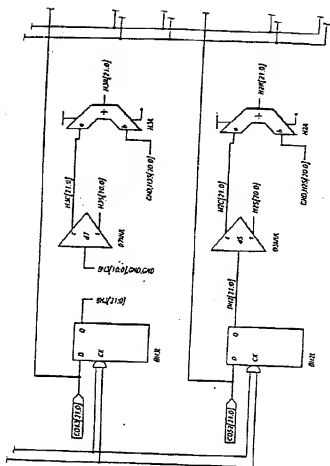


FIG. 15c

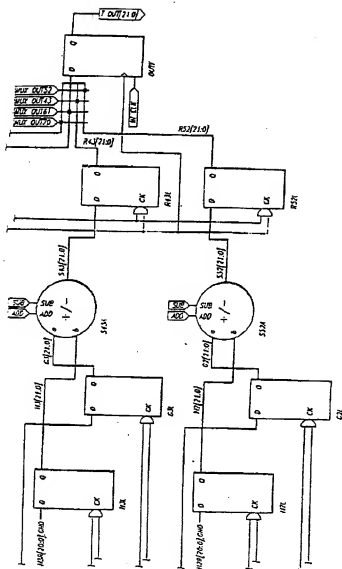
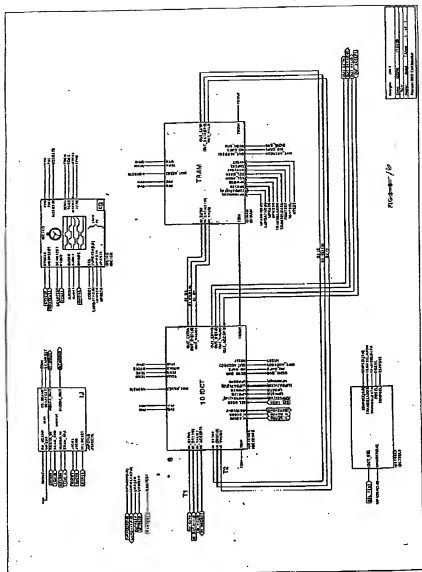
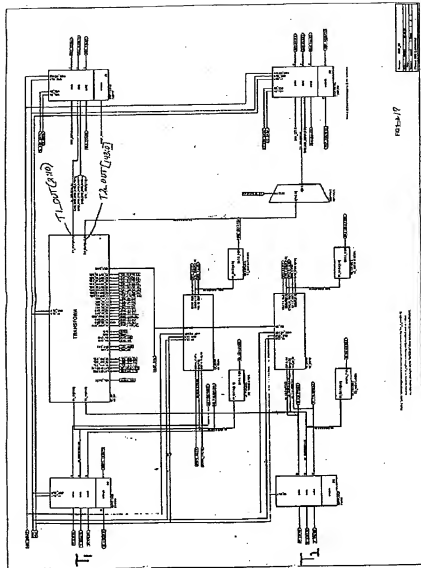
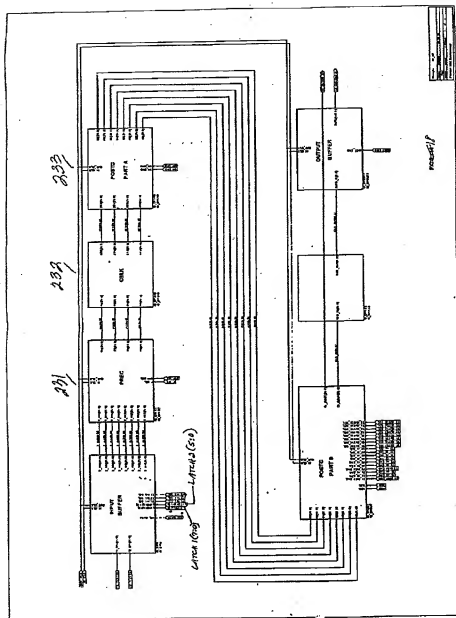
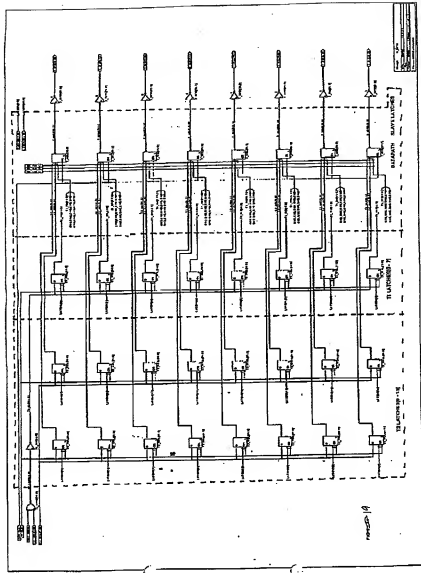


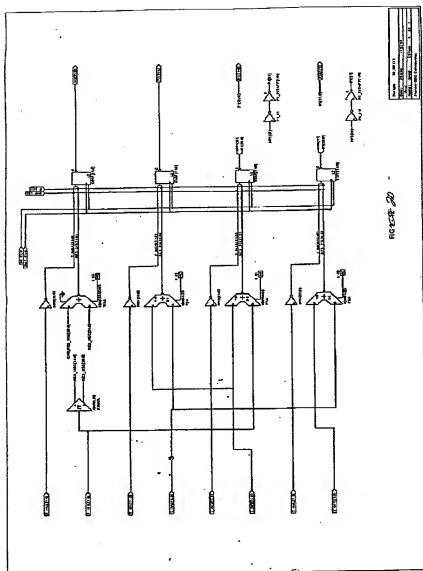
FIG. 15d



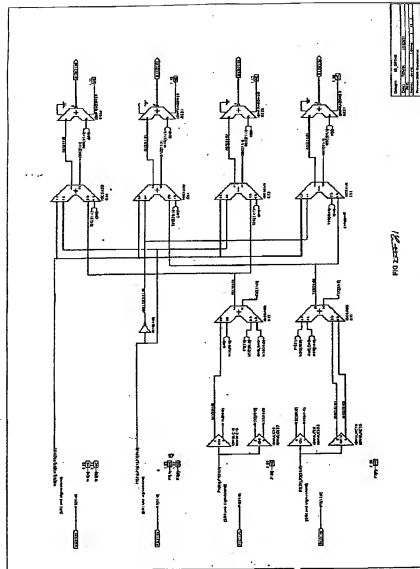


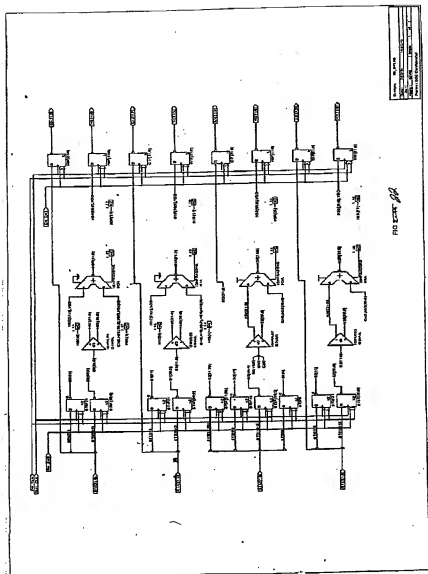


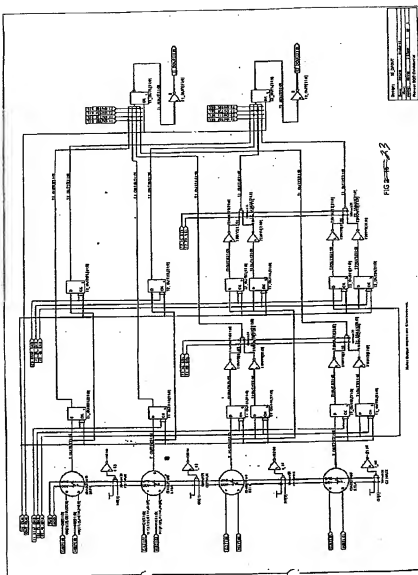


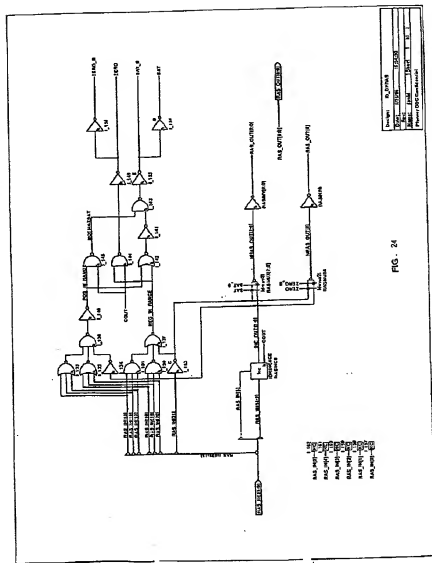


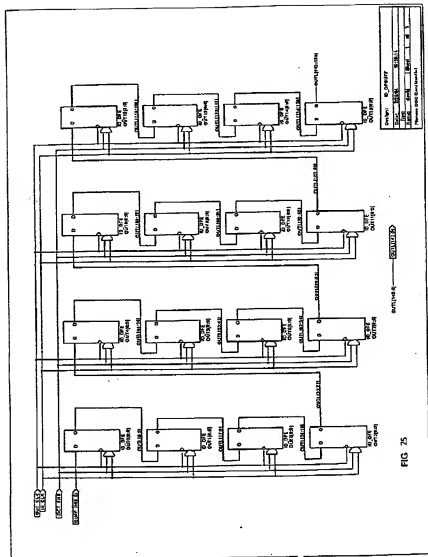




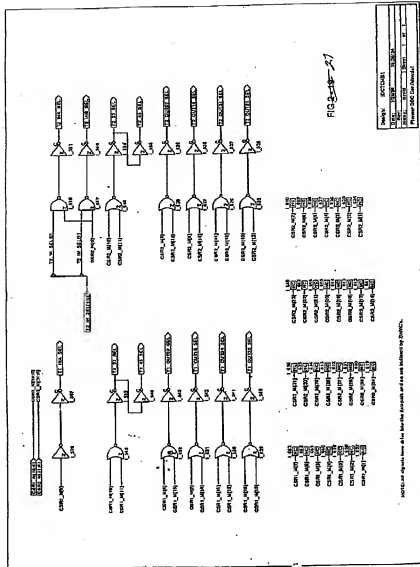


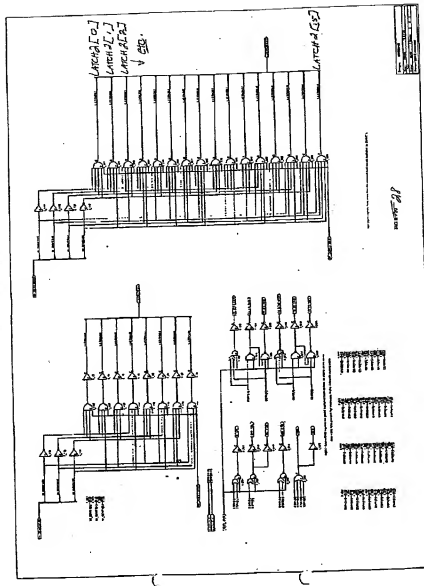




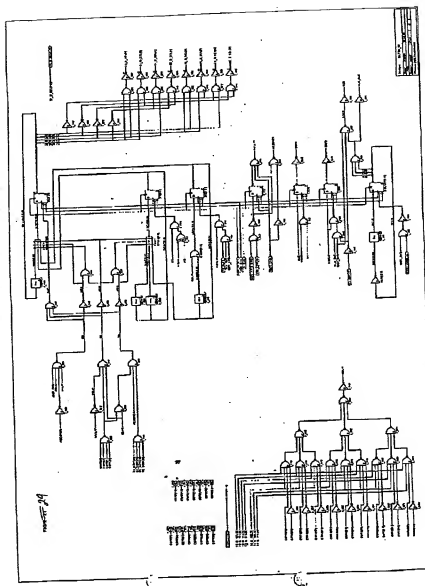


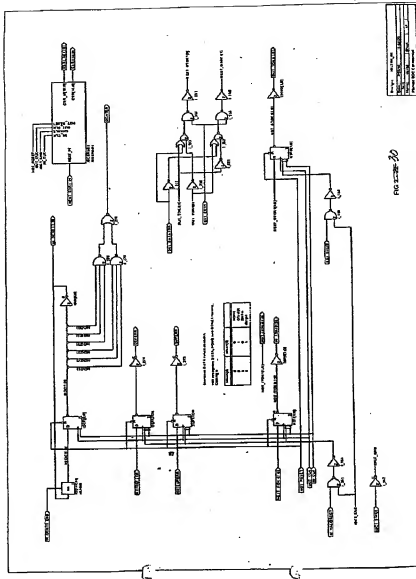


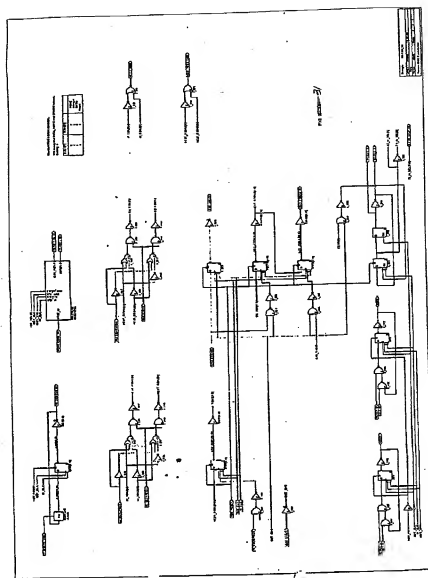






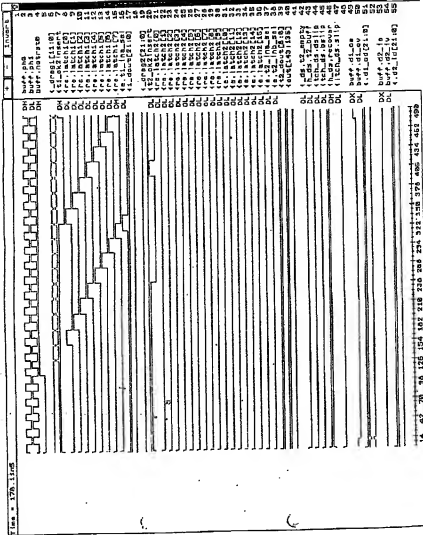




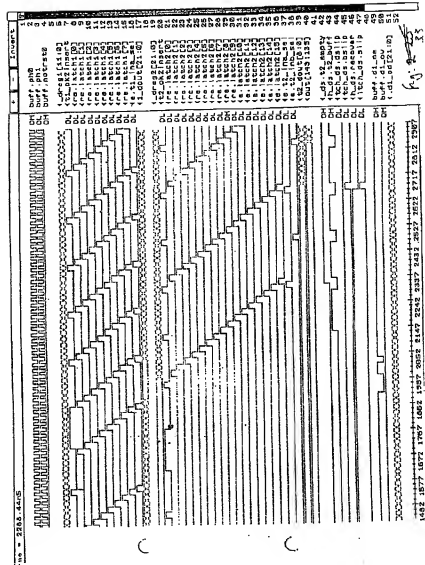


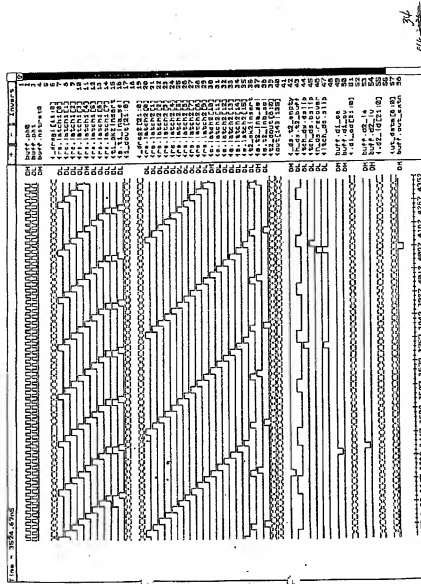
45

32

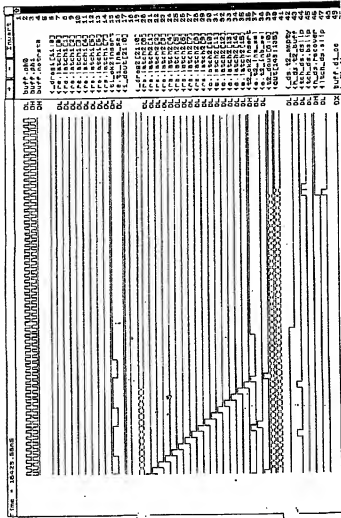


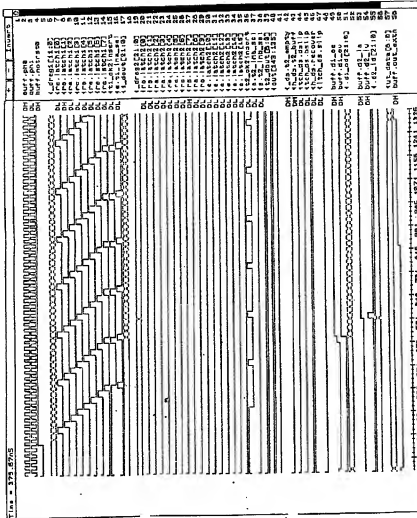
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95 1.00





38

38  
38





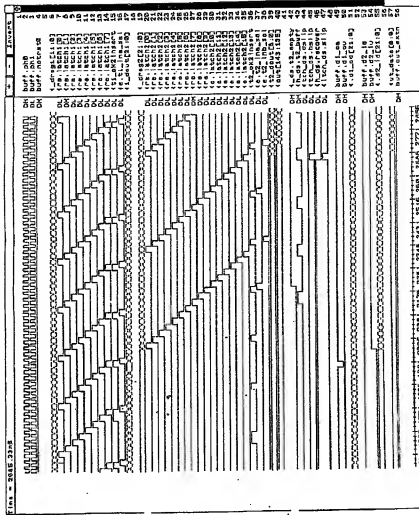
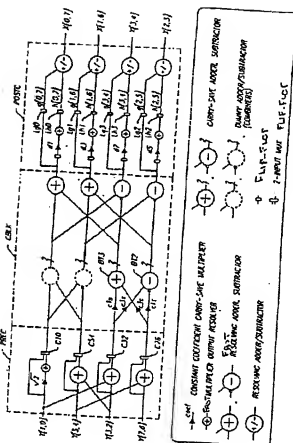


FIG. 38



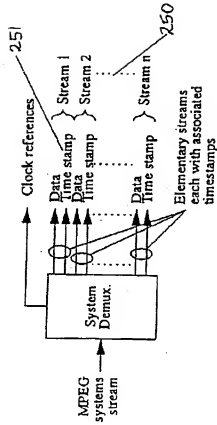


Figure 39

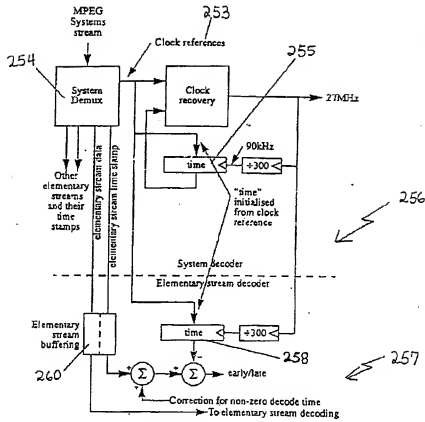


Figure 40

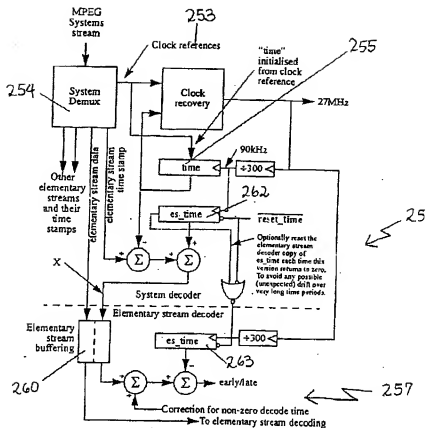


Figure 41

45

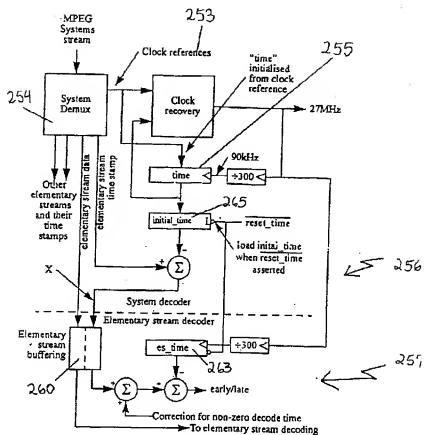


Figure 42

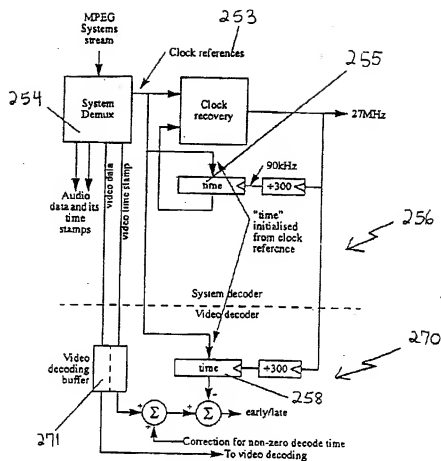
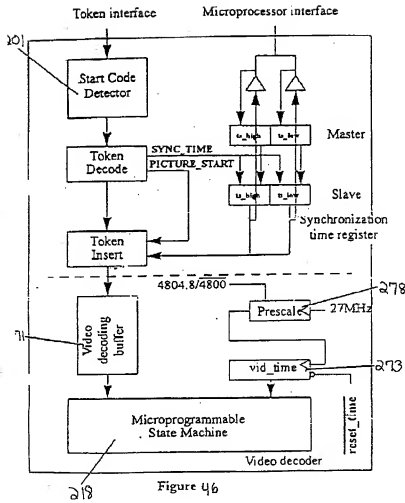


Figure 43









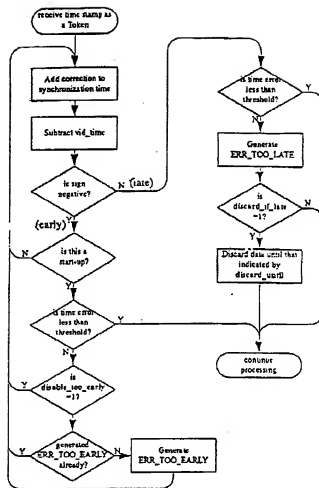


Figure 47

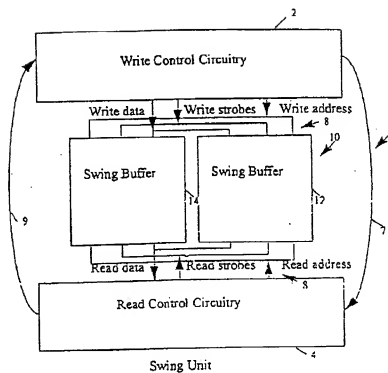


Figure 48

Swing Unit

42

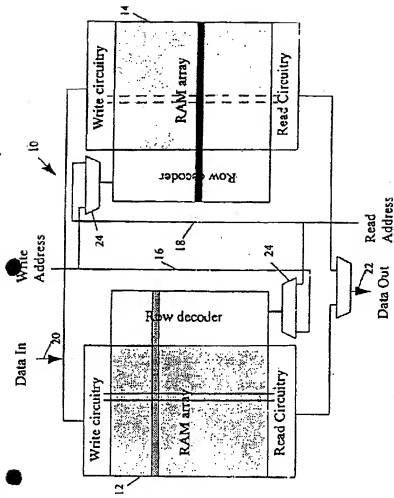


Figure 4q

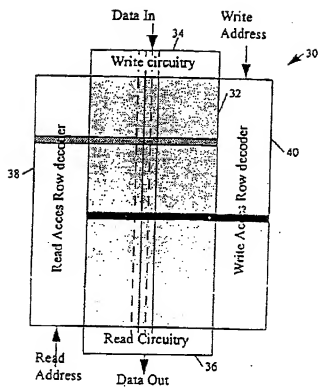


Figure 50



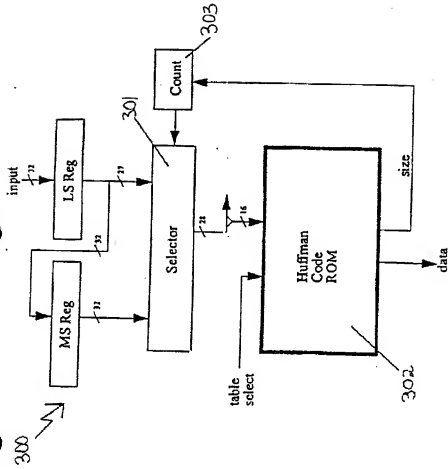
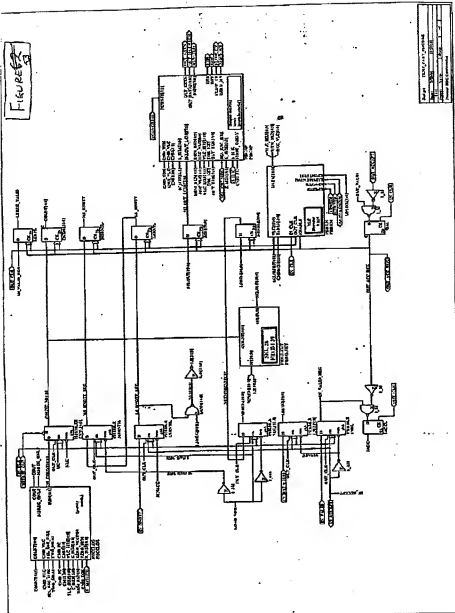


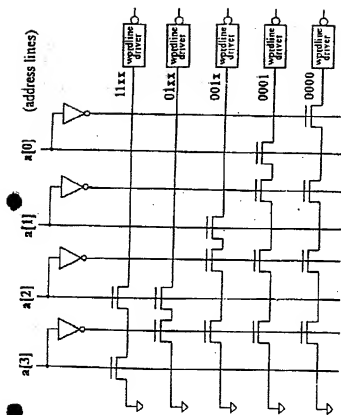
Figure 52







50



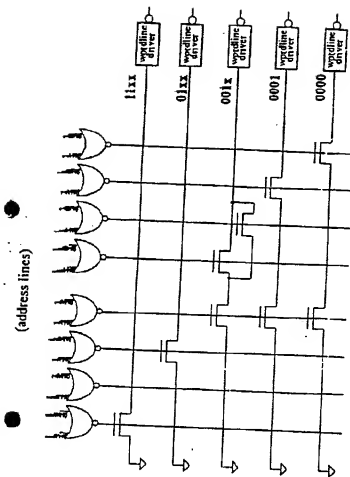


Figure 56

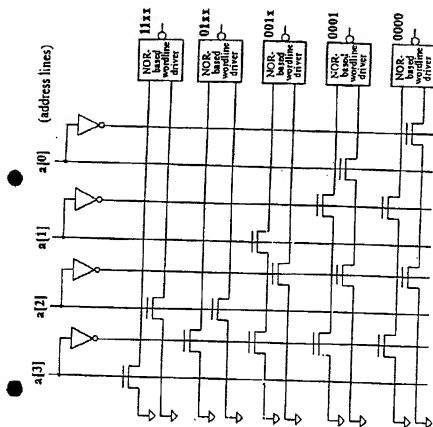


Figure 57

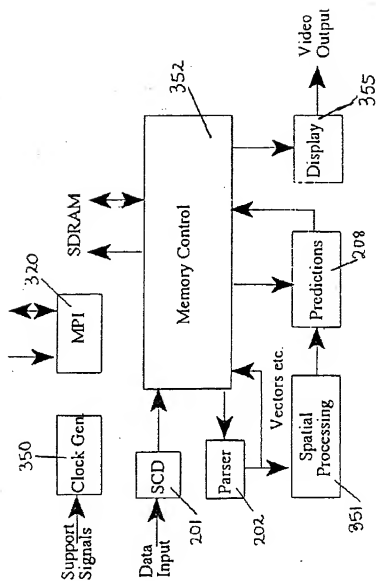


Figure 58

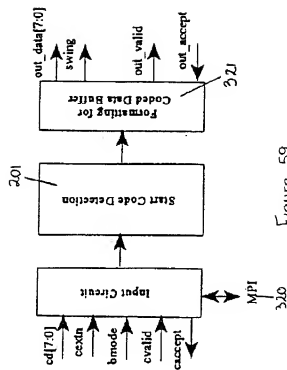


Figure 59

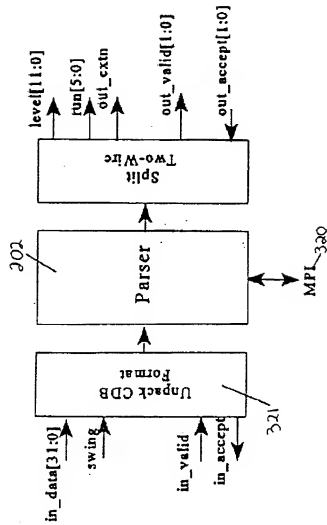


Figure 60



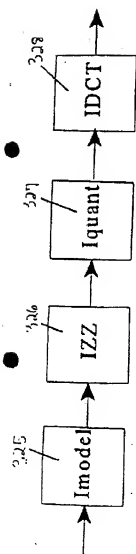


Figure 6

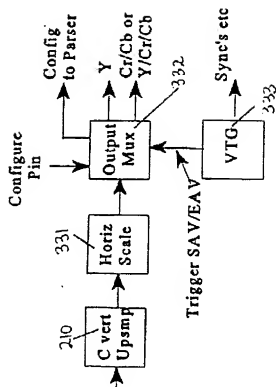


Figure 62

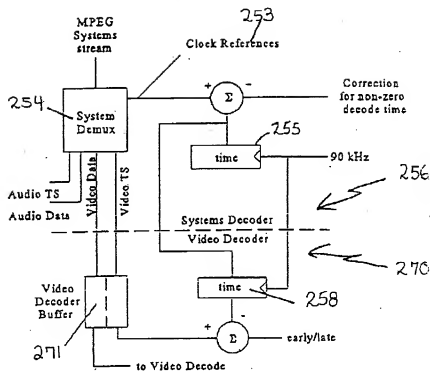


Figure 63

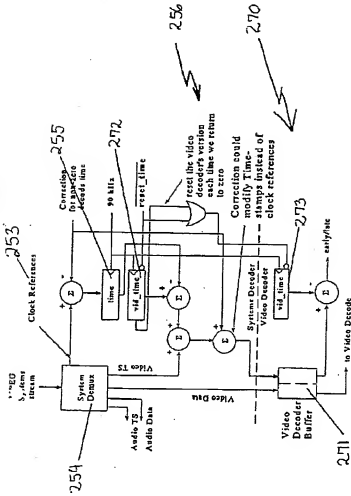


Figure 64

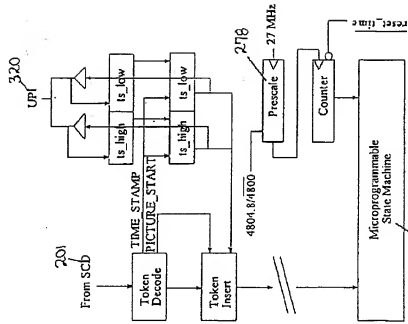


Figure 65

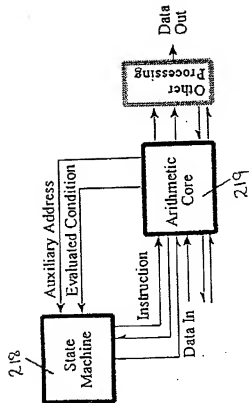


Figure 66

70

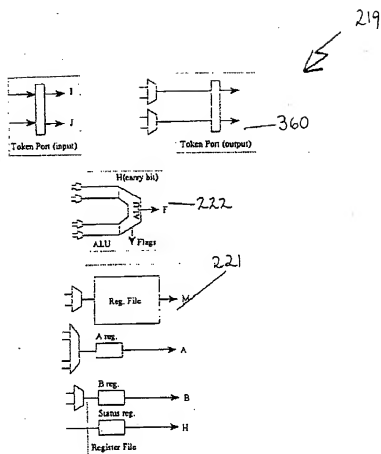


Figure 67

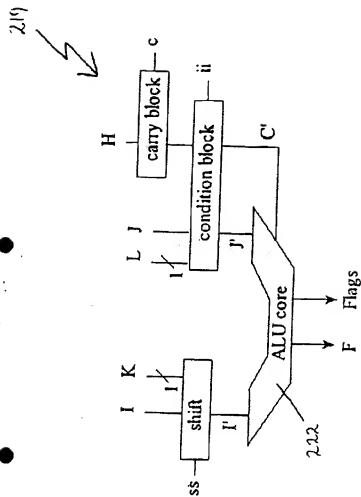


Figure 68



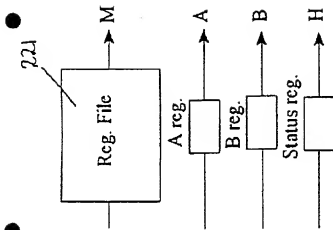


Figure 69

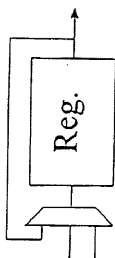


Figure 10

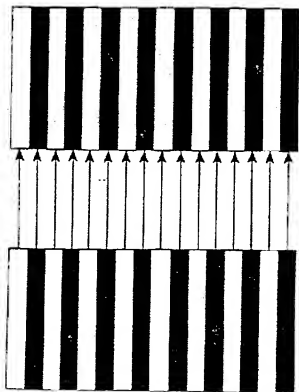


Figure 71

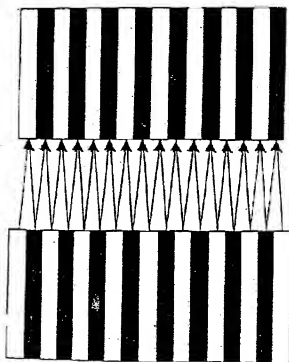


Figure 72

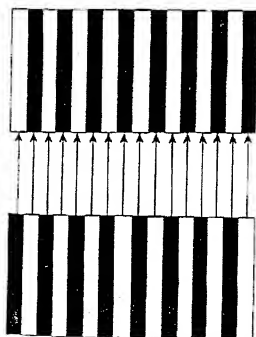


Figure 73

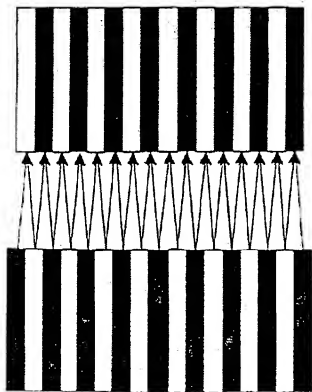


Figure 74

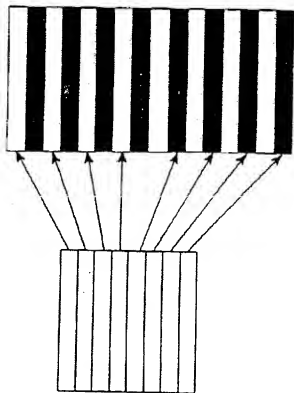


Figure 75

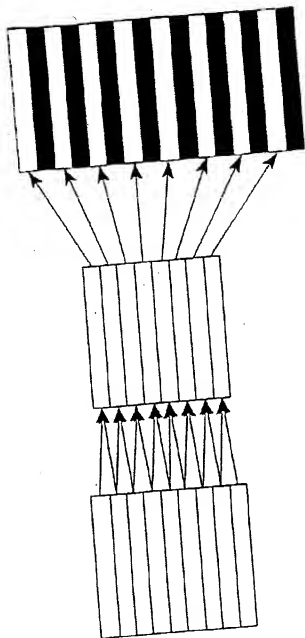


Figure 7c



Q:

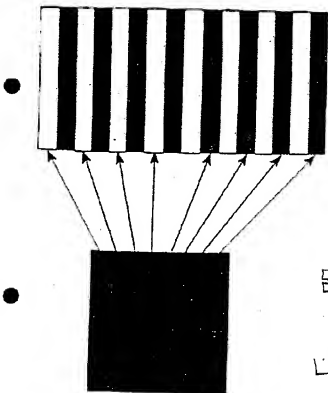


Figure 77

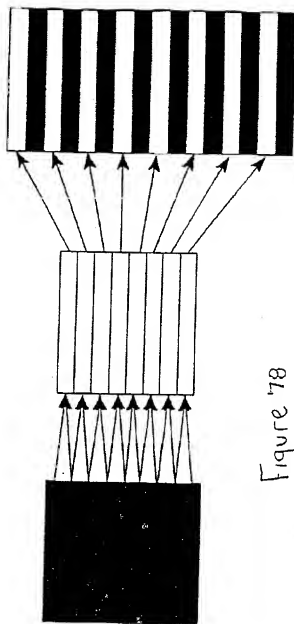


Figure '78

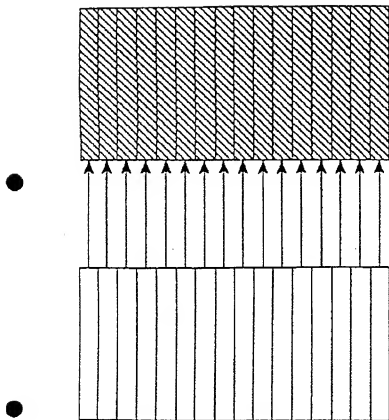


Figure 79

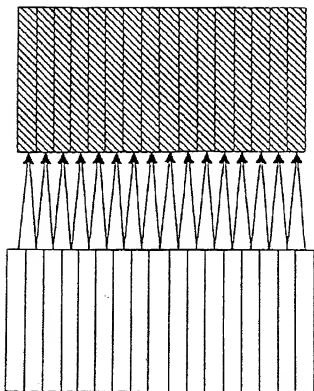


Figure 80

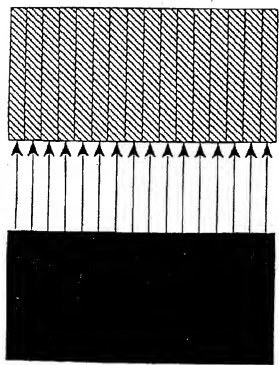


Figure 81

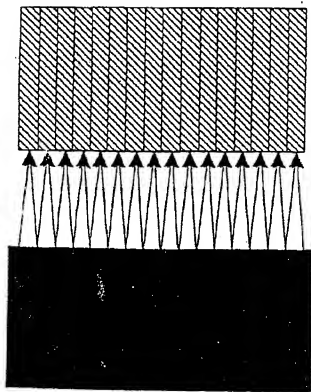


Figure 82

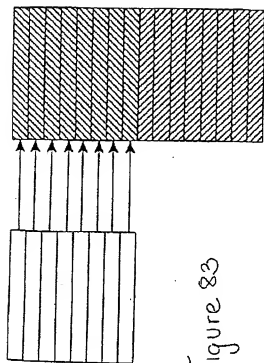


Figure 83

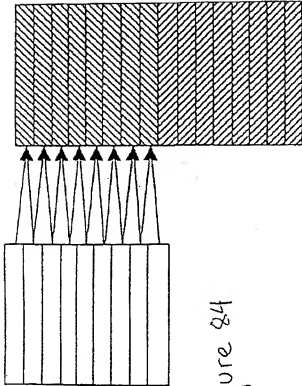


Figure 24



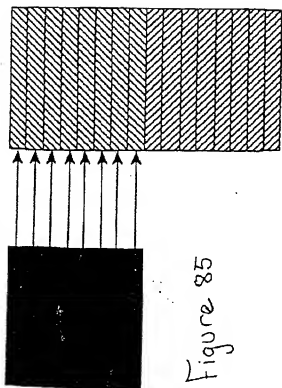


Figure 85

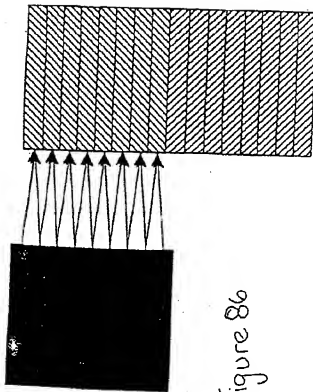


Figure 86

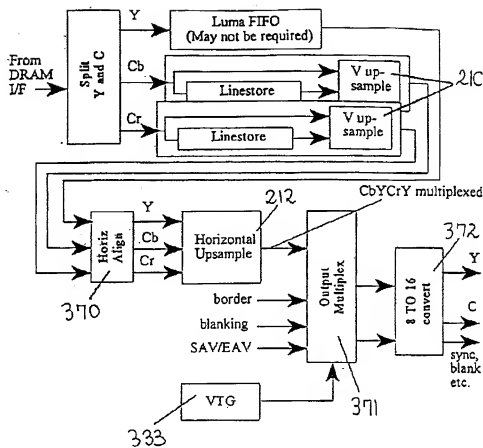


Figure 87

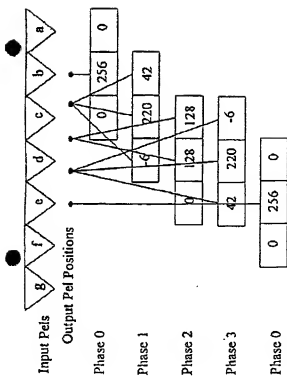


Figure 88

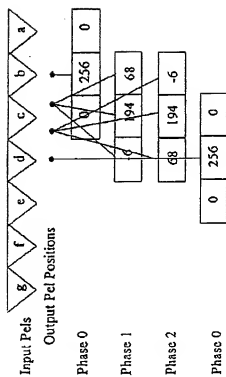


Figure 89

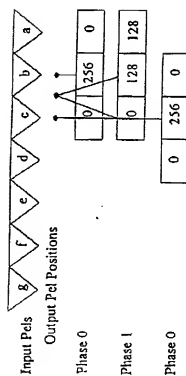


Figure 90

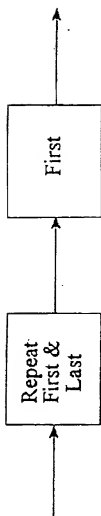


Figure 91

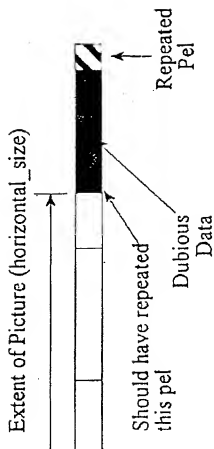


Figure 92



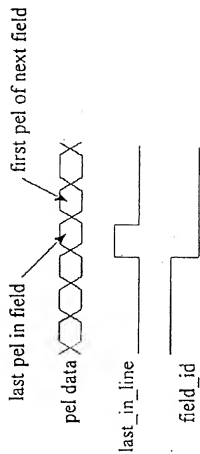


Figure 93

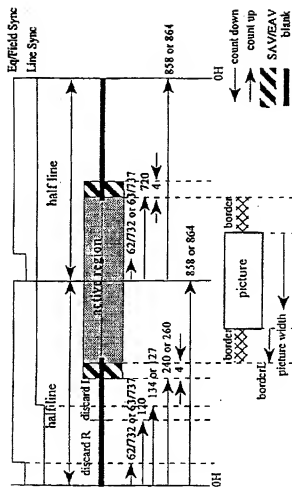


Figure 94

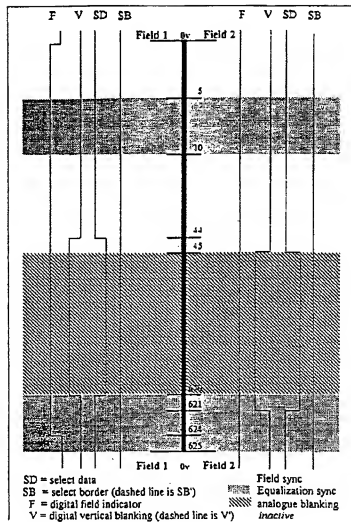


Figure 95

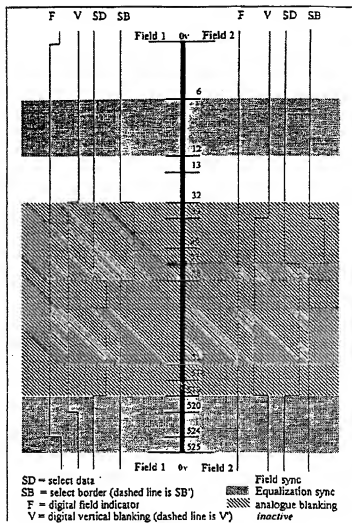


Figure 96

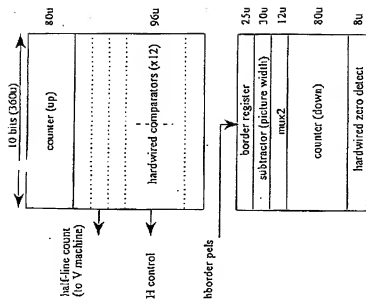


Figure 97

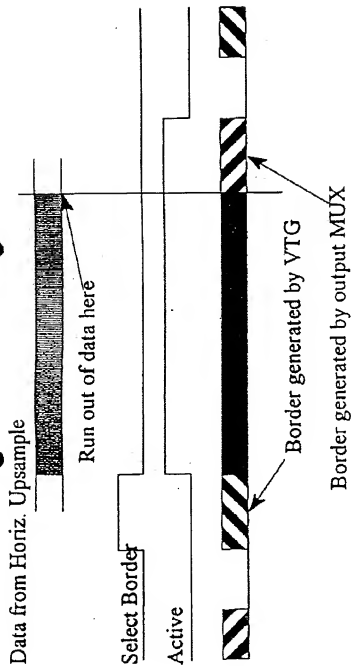


Figure 98

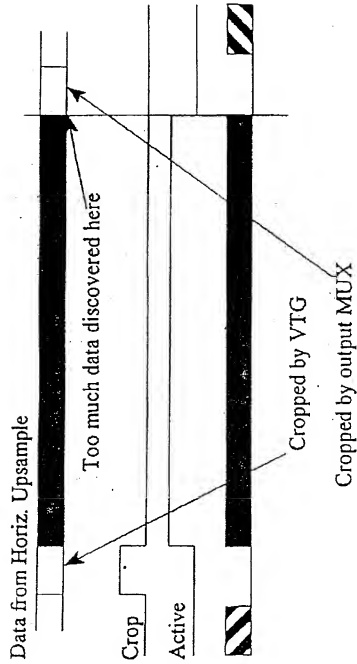


Figure 99

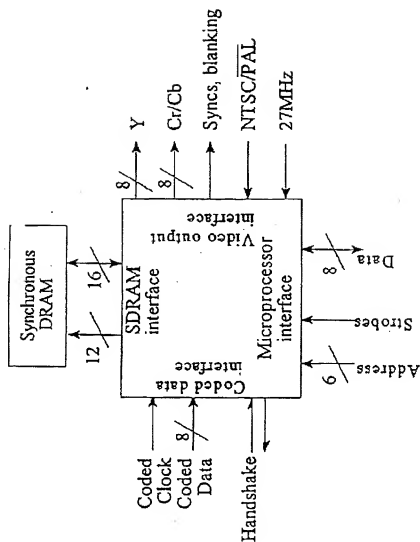


Figure 100



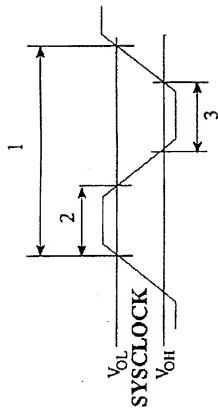
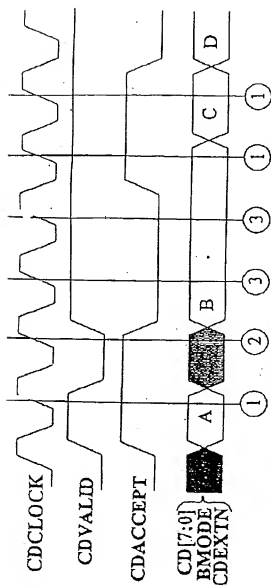


Figure 101



- 1) CDVALID and CDACCEP both HIGH - data transfer occurs.
- 2) CDVALID is LOW - SYSTEM ignores data.
- 3) CDACCEP is LOW - SYSTEM cannot accept data, data is present again until accepted

Figure 102

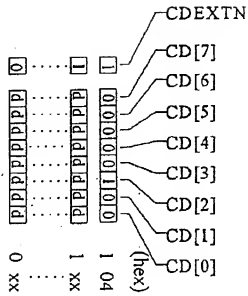
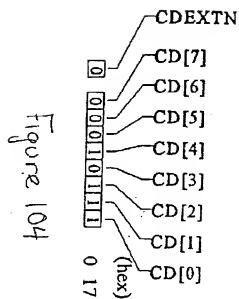


Figure 103



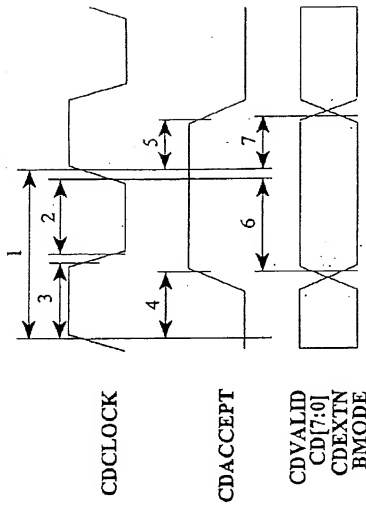


Figure 105

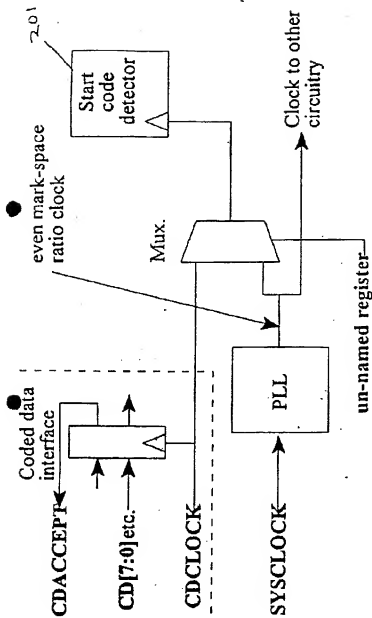


Figure 106

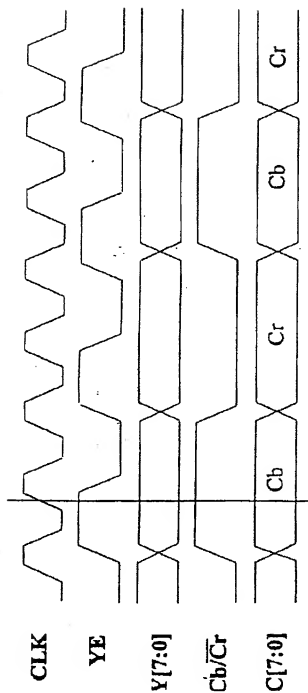


Figure 107

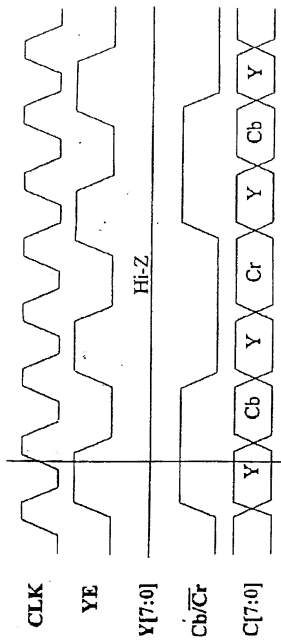


Figure 108



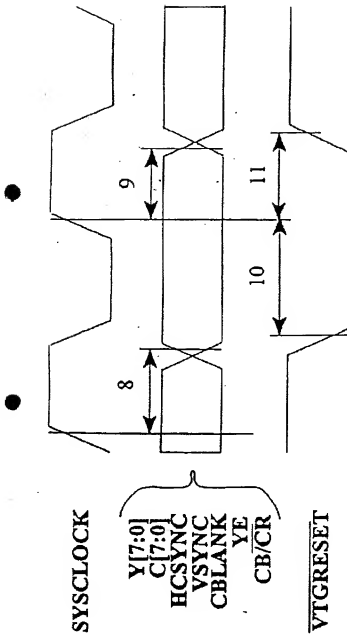


Figure 109

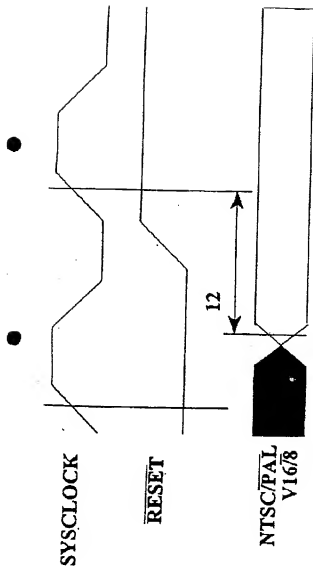


Figure 110

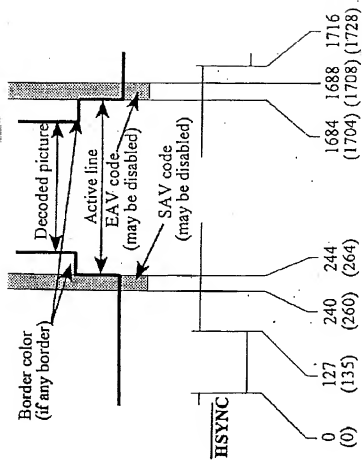
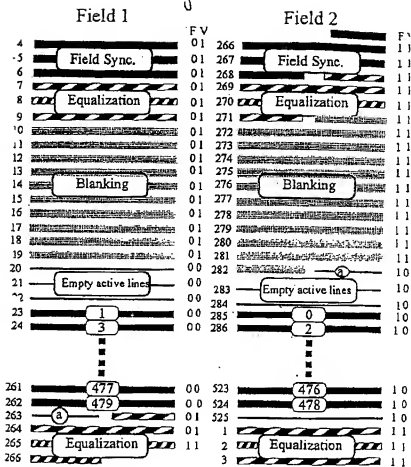
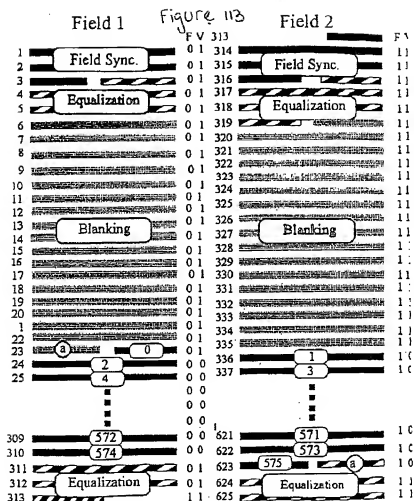


Figure III

Figure 112.



a: CBLANK is inactive but Y[7:0] still output blanking



a: CBLANK is inactive but Y[7:0] still output blanking

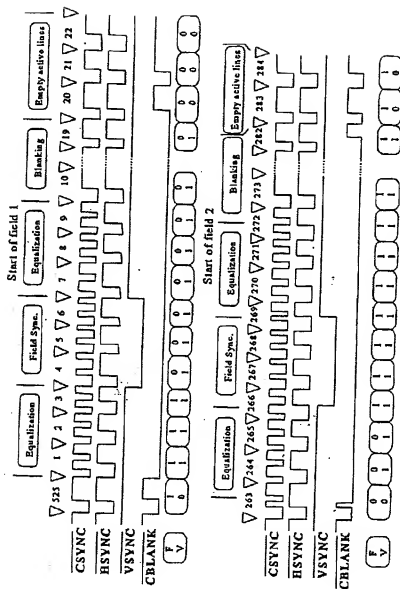
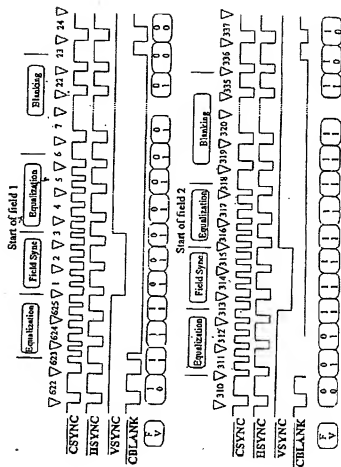


Figure 114

Figure 115



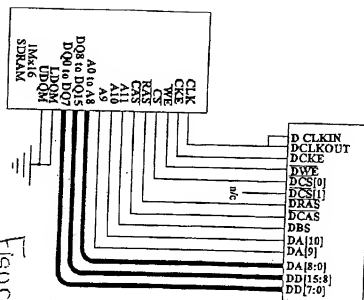


Figure 11b



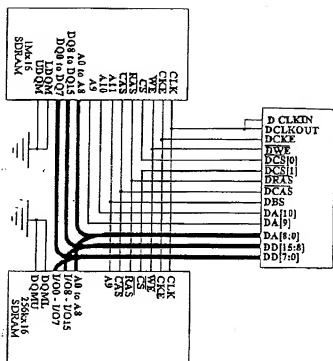


Figure 117

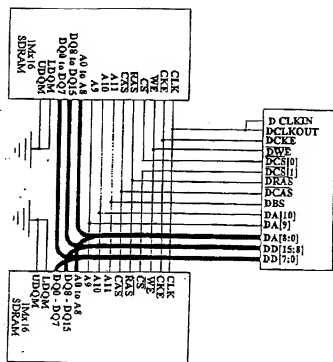


Figure 118

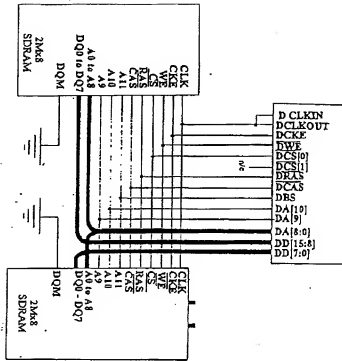


Figure 119

/23

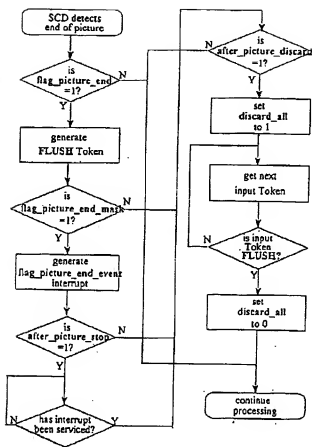


Figure 120

124

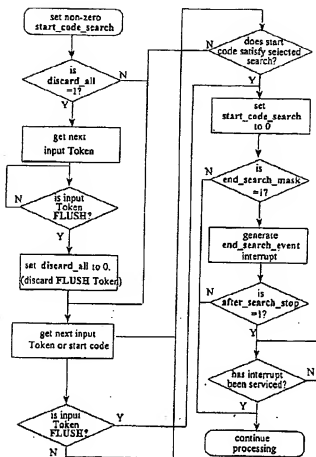


Figure 121

125

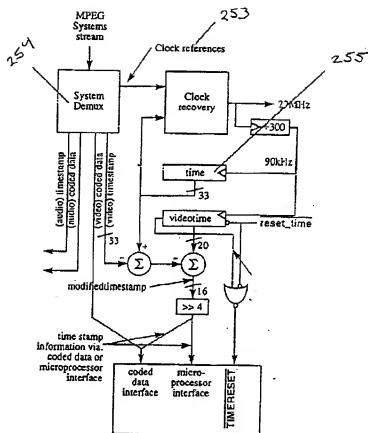
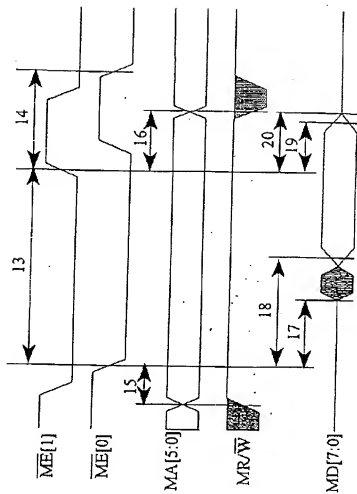


Figure 122

126

figure 123



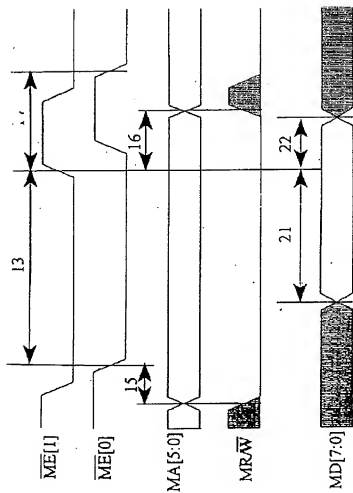


Figure 124



**ABSTRACT**

An MPEG video decompression method and apparatus utilizing a plurality of stages interconnected by a two-wire interface arranged as a pipeline processing machine. Control tokens and DATA Tokens pass over the single two-wire interface for carrying both control and data in token format. A token decode circuit is positioned in certain of the stages for recognizing certain of the tokens as control tokens pertinent to that stage and for passing unrecognized control tokens along the pipeline. Reconfiguration processing circuits are positioned in selected stages and are responsive to a recognized control token for reconfiguring such stage to handle an identified DATA Token. A wide variety of unique supporting subsystem circuitry and processing techniques are disclosed for implementing the system, including memory addressing, transforming data using a common processing block, time synchronization, asynchronous swing buffering, storing of video information, a parallel Huffman decoder, and the like.